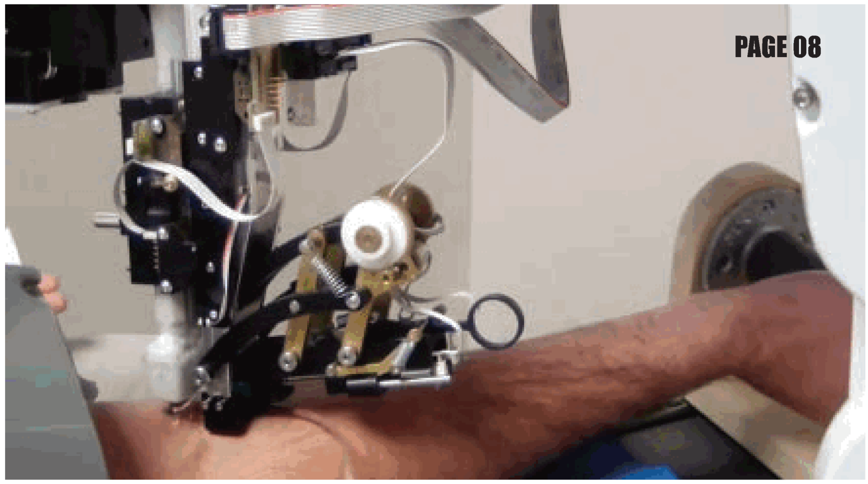

Design • Build • Program • Learn • Compete
SERVO
SERVO
FOR THE ROBOT INNOVATOR
www.servomagazine.com
MAGAZINE
October 2013
The Evolution of
LEGO
From RCX to EV3
U.S. $5.50 CANADA $7.00
10
0 71486 02422 4


Pololu
Robotics & Electronics
Finding the right parts for your robot can be difficult, but you also don't want to spend all your time reinventing the wheel (or motor controller). That's where we come in: Pololu has the unique products — from actuators to wireless modules — that can help you take your robot from idea to reality.
Dual VNH5019 Motor Driver Shield for Arduino
ITEM #2502
$49.95
Control two brushed DC motors with an Arduino or Arduino-compatible board. Operates from 5.5 V to 24 V and can deliver a continuous 12 A (30 A peak) per motor.
Passive Infrared (PIR) Detector SE-10
ITEM #1635
$9.95
• Great for motion detection
• 5 V to 12 V operation
37D mm Metal Gearmotors and Machined Aluminum Bracket
$24.95
Many other gearmotors available
ITEM #1995
$7.95
We stock these powerful gearmotors in several gear ratios with optional integrated encoders. Mount them securely to your project with our light and sturdy machined aluminum bracket.
Pololu Carrier with Sharp Digital Distance Sensor
ITEM #1134
$6.95
• Detects objects between 2 cm and 10 cm away
• Built-in indicator LED
Addressable RGB 30-LED Strip, 1 m
ITEM #2540
$34.95
Waterproof, individually addressable RGB LED strip that runs on 5 V. Can be chained to form longer strips or cut for shorter sections. Also available in 2 m and 5 m lengths.
5 V Step-Up Voltage Regulator U1V10F5
ITEM #2564
$4.49
3.3 V version available
• 0.5 V to 5.5 V input
• Linear down-regulation
• 70% to 90% efficiency
Force-Sensing Resistor
ITEM #1696
$6.95
Add a touch interface to your project with this 0.5″ diameter force sensing resistor. Can sense an applied force ranging from a few dozen grams to over 10 kg.
Orangutan SVP-1284 Robot Controller
ITEM #1327
$99.95
Robot controller with user-programmable ATmega1284, integrated programmer, USB interface, motor drivers, 21 free I/O lines, quadrature encoder decoder, LCD, buzzer, pushbuttons, and more.
NiMH Battery Packs and iMAX-B6AC Balance Charger and Discharger
STARTING AT
$4.49
ITEM #2260
$49.95
Single cells and battery holders also available
QTR-3RC Reflectance Sensor Array
ITEM #2457
$4.95
• Great for line-following robots
• Light weight: 0.6 g
• Small size: 1.25″ x 0.3″
Crimping Tool
ITEM #1928
$34.95
26 AWG Red Stranded Wire, 70 ft
ITEM #2622
$4.50
Other colors and gauges available
Crimp Connector Housings
STARTING AT
59¢/pack
Male Crimp Pins
ITEM #1931
$7.95/100-pack
Female Crimp Pins
ITEM #1930
$5.95/100-pack
16-26 AWG Wire Stripper
ITEM #161
$6.95
Everything you need to make custom cables!
Zumo Robot for Arduino
(Assembled with 75:1 HP Motors)
ITEM #2506
$99.95
Build and customize your Zumo!
98 mm
98 mm
Individual parts and kit version also available — build your own configuration!
• Two micro metal gearmotors coupled to a pair of silicone tracks
• Stainless steel bulldozer-style blade
• Array of six infrared reflectance sensors for line following or edge detection
• 3-axis accelerometer and magnetometer
• Buzzer for simple sounds and music
Find these products and more at: www.pololu.com

SERVO MAGAZINE

10.2013

VOL. 11 NO. 10



Columns

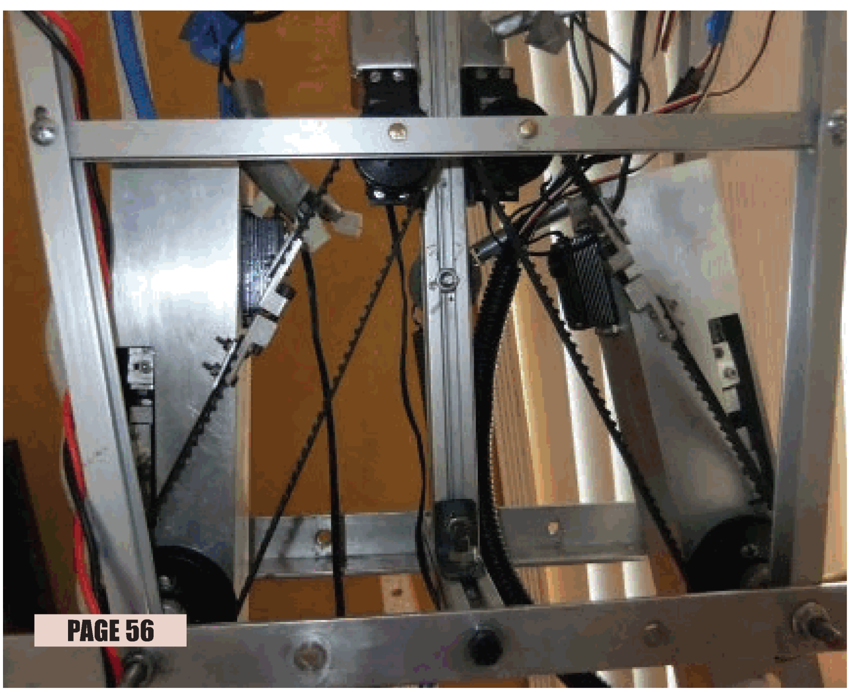
The Combat Zone...

Departments

SERVO Magazine **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

In This Issue ...

PAGE 38



PAGE 56

Mind / Iron

by Bryan Bergeron, Editor

Self-Replicating Robots — Sort Of

Robots capable of building themselves — think the self-replicating nanobots or Replicators in Stargate Atlantis — are here. Well, sort of. Armed with an affordable 3D printer (a.k.a., extrusion robot), a talented experimentalist can crank out the mechanical assembly for basic robot platforms. Batteries, servos, and electronics are still a ways off, but the basics are there. We've achieved the ability to crank out robots using local resources, albeit not as originally envisioned.

When I bought my 3D printer, I thought that having one on my desk would help me stay abreast of what's happening in the field. Well, I was wrong. True, I can get hands-on experience with different rendering/extrusion/finishing programs and devices, but the field is just exploding too rapidly for me to follow everything and get anything else done.

In the medical arena, 3D printer robots are being used to create cranial implants, replacements, temporary teeth, artificial limbs, and orthodontics. Then, there's the whole world of medical 3D scanning, where robots are used to create scans for better fitting hearing aids, replacement teeth, and models to preview the results of reconstructive surgery.

On the fun front, self-replication has had a boost from announcements and demonstrations of affordable desktop 3D scanners. If you're a wiz at using a 3D authoring program, then you may not need a scanner. However, let's say a support on your robotics platform is cracked and needs to be replaced. What could be easier than simply scanning the support, using a touch-up tool to remove the crack in the digital image, and then sending the file to your 3D printer.

Still don't have a 3D printer on your desk? It's becoming less of an issue, thanks to increased availability of web-based 3D printing services. If you've ever designed a printed circuit board and ordered the finished board from one of the online services, then you'll be at home having your 3D printing outsourced. I'm waiting for my local Kinko's to offer 3D printing along with their 2D printing service.

At home, printing isn't for everyone, but more activity in 3D printing makes it better for everyone. An increase in the popularity of 3D printing will attract software and hardware developers to the space. There will eventually be an 'app for that.'

Back to self-replicating robots, what can the current generation of affordable robotic extruders do? Quite a bit. Search for "robot" on Thingiverse (**www.thingiverse.com**) — the 3D image repository — and you'll find 3D files to create everything from quadcopters, biped walkers, and robot arms to multi-legged insectoid robots. Just add electronics, batteries, and a few nuts and bolts, and after extruding a few spools of PLA or ABS plastic, you'll have an army of robots.

Not quite self-replicating, but in this scenario, a robot is semi-autonomously replicating another robot. Sort of like Escher's Drawing Hands, where one hand is sketching the other. **SV**

FOR THE ROBOT INNOVATOR **SERVO** MAGAZINE

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct., Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore Only **1-800-783-4624**
www.servomagazine.com

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277, N. Hollywood, CA 91615

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
display@servomagazine.com

EDITOR
Bryan Bergeron
techedit-servo@yahoo.com

CONTRIBUTING EDITORS
Jeff Eckert, Jenn Eckert, Tom Carroll, Kevin Berry, Dennis Clark, R. Steven Rainwater, Michael Simpson, Gordon McComb, Bryce Woolley, Evan Woolley, Steven Nelson, John Blankenship, Samuel Mishal, Daniel Albert, Chris Mayer, Ray Billings, Zac O'Donnell, Chris Olin, Tiffany Clegg

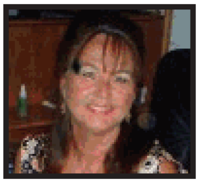
CIRCULATION DEPARTMENT
subscribe@servomagazine.com

MARKETING COORDINATOR WEBSTORE
Brian Kirkpatrick
sales@servomagazine.com

WEB CONTENT
Michael Kaudze
website@servomagazine.com

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

PRODUCTION
Sean Lemieux

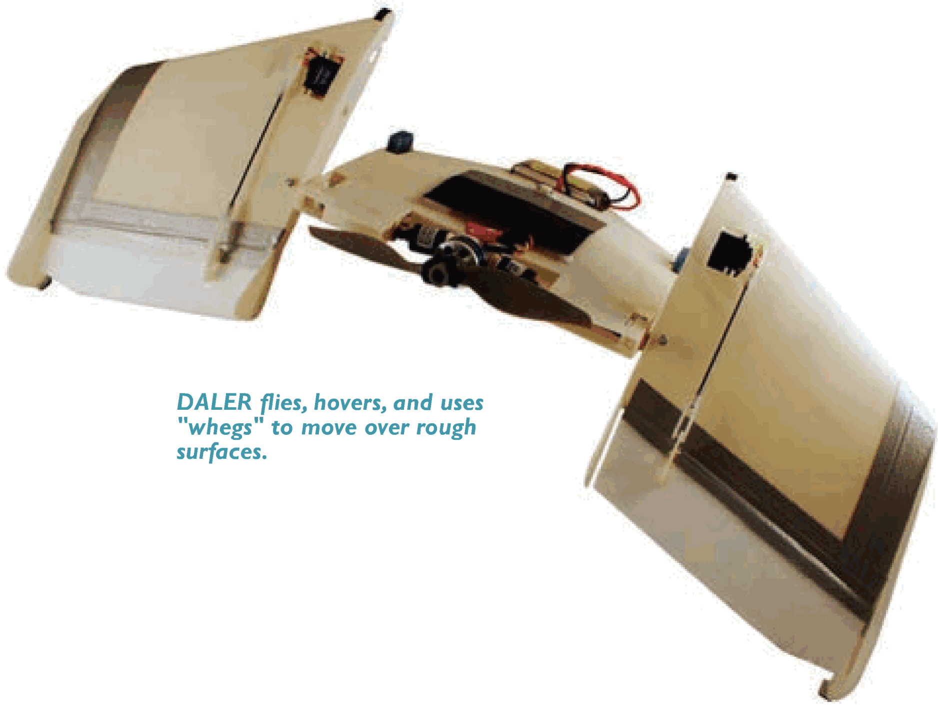



Robytes

by Jeff and Jenn Eckert

Robot Flies and Walks

Most flying bots are equipped with some sort of landing gear — typically wheels — so they can move along the ground when necessary. Unfortunately, that adds to the structural weight, and simple wheels aren't particularly good at navigating over rough terrain. Never fear, however, because the folks at EPFL's Laboratory of Intelligent Systems (**lis.epfl.ch**) have come up with the Deployable Air Land Exploration Robot (DALER) which incorporates "adaptive morphology" to repurpose its wings for walking. These wing-legs ("whegs") are designed to manage more complex terrains, including semi-collapsed buildings, caverns, and forests. This is intended to make DALERs useful for applications like search-and-rescue, environmental monitoring, and other tasks in which they are able to fly, hover, and crawl as required. According to its developers, "The morphology of the robot is optimized for ground speed. It can move forward at 0.2 m/s (8 in/s) and can rotate on spot at 25°/s. The robot is capable of walking with different gaits, it can move on different surfaces, it can overcome high obstacles, and it can also navigate in rough terrains." Its gait is not exactly graceful, as you can see in a YouTube video (just search "daler robot"). Plus, in its present form at least, you toss DALER like a Frisbee to launch it, so it doesn't appear to be capable of becoming airborne on its own. I'm sure they'll work that out.



DALER flies, hovers, and uses "whegs" to move over rough surfaces.

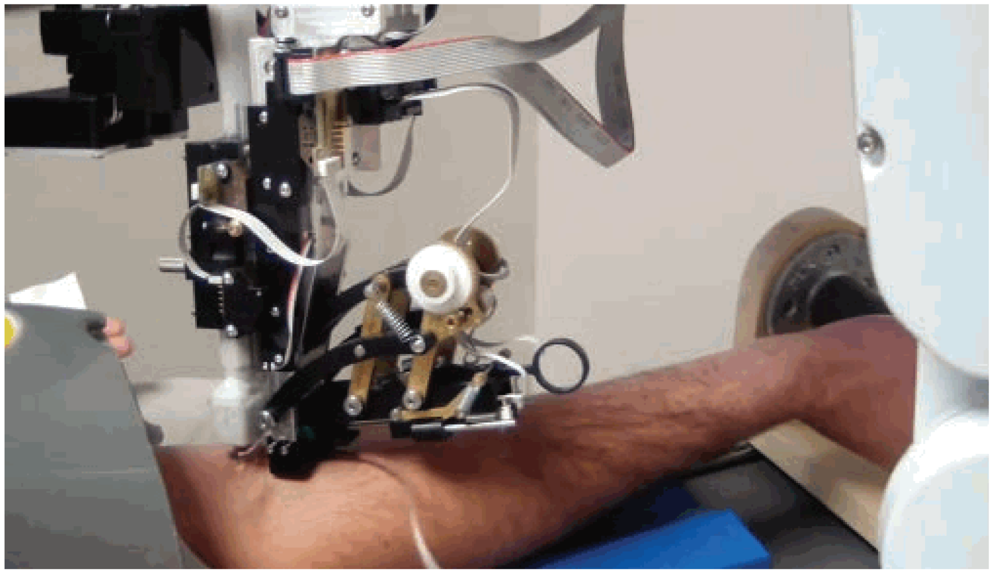
The new X-RHex Lite climbs easily over many obstacles.

Hexapod Lite Debuts

Designed to handle much more challenging obstacles is RHex, which is short for "robot hexapod." This hopperbot was actually developed a decade ago in a multi-university collaboration, but the latest version — X-RHex Lite (a.k.a., XRL) — is lighter and more agile. Recently introduced by the University of Pennsylvania's Kod*lab, it "can execute double jumps, flips, and — through a combination of moves — even pull-ups. For the tallest obstacles, the robot can launch itself vertically, hook its front legs on the edge of the object it's trying to surmount, then drag its body up and over." The ultimate aim is to create a machine that can easily flip over rubble in a rescue mission or collect environmental data while crossing the desert. A highly entertaining 2.5 minute video is posted at **kodlab.seas.upenn.edu**.

Go to **www.servomagazine.com/index.php?/magazine/article/october2013_Robytes** to comment on these topics.

Vampire Bot Developed



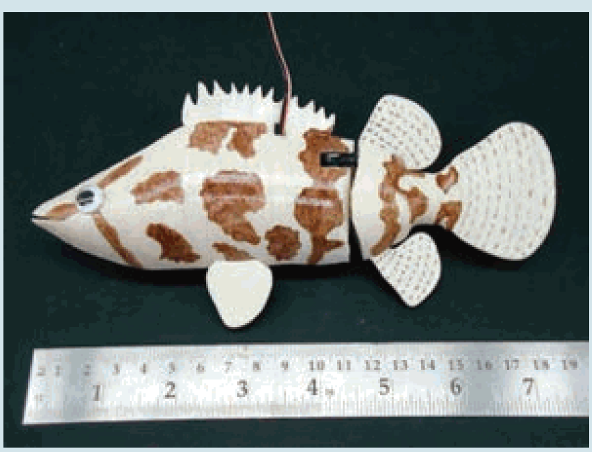
The Veebot venipuncture machine aims for more accurate blood draws.

If you like the idea of having your blood sucked out by a machine instead of a human phlebotomist, you'll love the Veebot robot developed by Veebot LLC (**www.veebot.com**) — a startup company in Mountain View, CA. The company's mission is to "make the whole procedure of venipuncture automated to reduce error and decrease venipuncture times. This saves hospitals and clinics money, reduces the risk of injury to practitioners, and improves comfort and care for patients."

So far, the machine identifies the best vein to poke only about 83 percent of the time, which is no better than a human can do. The company expects to boost it up to 90 percent before beginning clinical trials in a few months. The big question is whether patients will get beyond the creep factor.

Beer for Me, Vodka for the Fish

As every lounge lizard knows, alcohol is highly useful for dissipating a potential victim's natural fear and anxiety. It seems to work pretty much the same way in fish, according to a paper published in *PLOS ONE*, an international peer-reviewed online publication (**www.plosone.org**). Professors Maurizio Porfiri and Simone Macri revealed the details of an experiment in which some real live zebrafish were placed in a three-chambered tank with a robotic version of an Indian leaf fish — a natural predator of the zebrafish. The control (i.e., sober) group avoided the robot completely and stayed in a separate chamber. When other groups of fish were exposed to ethanol, they no longer bothered to avoid the bot.



Robotic Indian leaf fish — scary only if you're sober.

In another test, the profs divided a tank into light and dark compartments and created a simulated heron attack from the tank's surface. The sober fish preferred the lighter part of the tank and successfully avoided the attacker. The drunk ones, however, hung out in the dark part and were slow to swim away when attacked.

The abstract doesn't touch on any practical applications for this knowledge, but if you have ever had the urge to take a carp up to your hotel room, this could be helpful. In case you were concerned, "Acute administration of ethanol causes no harm and has no lasting effect on zebrafish." So back off, PETA. **SV**

ASK MR. ROBOTO

by Dennis Clark

Our resident expert on all things robotic is merely an email away.

roboto@servomagazine.com

Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?

Go to **www.servomagazine.com/index.php?/magazine/article/october2013_MrRoboto** to comment on this article.

It seems like every week there is breaking news from a Japanese or Korean lab dealing with walking bipedal robots. Awesome! In the US, there is news from "leg labs" from time to time, and more details about military robot "horses" that carry loads and sound like large leaf-blowers chasing you. (Is that a prelude to some kind of movie or perhaps nightmare?) If you've come to the conclusion that I'm a bit biased towards more mundane uses for robots, you'd be right. Robot assistants for people who have lost the ability to walk or use their arms, or could use a little help around the house seem far more worthy endeavors than new, improved ways to kill people. However, just to prove I'm not completely consistent, I *love* the new RoboOne competitions with robot Sumo, wrestling, and boxing matches! Our question this month is from a reader who appears to be thinking along these same lines ...

Q. I want to make a walking robot. I've looked at hobby servos and they don't seem to be a good match for this project. I want some kind of feedback so that I can balance while walking, and I don't want a huge bundle of wires. Is this possible? Are there servos that can give me that information?

— Bobby via email

A. Bobby, I like where you are going with this, and yes, there are servos that can offer a great deal of feedback on not only their position, but current draw, heating, and other useful data. Let's look at a few! (There are more, but the ones I'll discuss are the servos that I found information about.)

First off, let's look at hobby servos and see what they offer. Hmm, not much. While Futaba and Hitec both offer programmable servos, neither of these systems will provide any feedback that can be used to check position or operating conditions.

My research also shows that only one of the major (and minor) makers of servos offers a contiguous bus configuration to minimize wiring. That one manufacturer is Futaba which has its S-bus receiver and servo system. The S-bus system allows S-bus servos to be programmed to know their location in the system.

Further, the servos reside along a single three-wire cable connected to the receiver. There are also S-bus translators that allow non-S-bus servos to be used in an S-bus system.

That's nice, but these servos are expensive and still don't provide position or force feedback.

Now, we'll move on to intelligent networked servos that you don't have to continually send pulses to in order for them to continue to function.

I have an admission to make. I know of three bus-networkable robot actuator systems. There may be more, but these are the only three that are talked about the most and that I have been able to find. They are comparable to each other in functionality, but each have their own unique capabilities.

They are the Robotis Dynamixel, RoboBuilder wCK, and Dongbu HerkuleX. The Dynamixel, HerkuleX, and wCK servos come in a variety of strengths and speeds. You can create a microcontroller interface to any of them (they all use standard UART protocols; either straight digital or RS485-style links.)

Before we go into any kind of details, here is where you can find programming information on all of these servos:

Dynamixel AX-12
www.trossenrobotics.com/images/productdownloads/AX-12%28English%29.pdf

Dongbu HerkuleX
www.robotshop.com/content/PDF/manual-drs-0101.pdf

RoboBuilder wCK
http://robosavvy.com/RoboSavvyPages/Robobuilder/robobuilder-creator-users-manual.pdf

Table 1 shows these servo actuator systems and their overall notable abilities. Each of these servo actuators have many more settings that you can configure, including the ability to get multiple servos to act at the same time, set motion limits, torque limits, and error conditions. It is a good evening's time in the arm chair of your choice to read through all of the possibilities.

Please note that there are several flavors of HerkuleX and Dynamixel servo actuators; each series may have different communications, power, and programming requirements. I've only noted manuals for the three types of servos that I have in my junk box.

All of these servos prices start out at around US$40. As you want stronger and faster actuators, the prices go up dramatically. I have included the wCK modules in this list for European readers because as far as I can tell, there are no current US distributors for these servos.

Servo actuators of this type are designed to be used in robots — not model cars and airplanes — so their connection systems are better tuned for us. For example, the Dynamixel actuator has mounting holes all over it for installing nut/bolt combinations (see **Figure 1**).

Not shown in the photo is the mounting hole on the back of the actuator that is in-line with the servo horn on the front. This allows the AX-12A to be fully supported at the joint for a strong connection.

The HerkuleX actuator (see **Figure 2**) does not have as many

Feature	HerkuleX DRS-0101	Dynamixel AX-12A	wCK-1111T
Voltage range	7-12V (7.4V rec.)	7-12V (9.6V rec.)	6-12V (9.6V rec.)
Stall torque[1]	12 kg-cm	16.5 kg-cm	11 kg-cm
Speed[2]	0.166 sec/60°	0.196 sec/60°	0.15 sec/60°
Communications	Full-duplex 8N1	Half-duplex 8N1	Full-duplex 8N1
Comm. Speed max	667 Kbps	1 Mbps	921 Kbps
Set transit speed	Yes	Yes	Yes
Set transit time	Yes	No	No
Set torque	Yes	Yes	Yes
Set LED	Yes (four LEDs)	Yes (one multi-color)	Yes (two colors)
Set temperature	Yes	Yes	No
Report position	Yes	Yes	Yes
Report loading	No	Yes	Yes

1. Stall torque at recommended voltage.
2. Speed at recommended voltage.

Table 1. Features of interest.

Figure 1.

Figure 2.

Figure 3A.

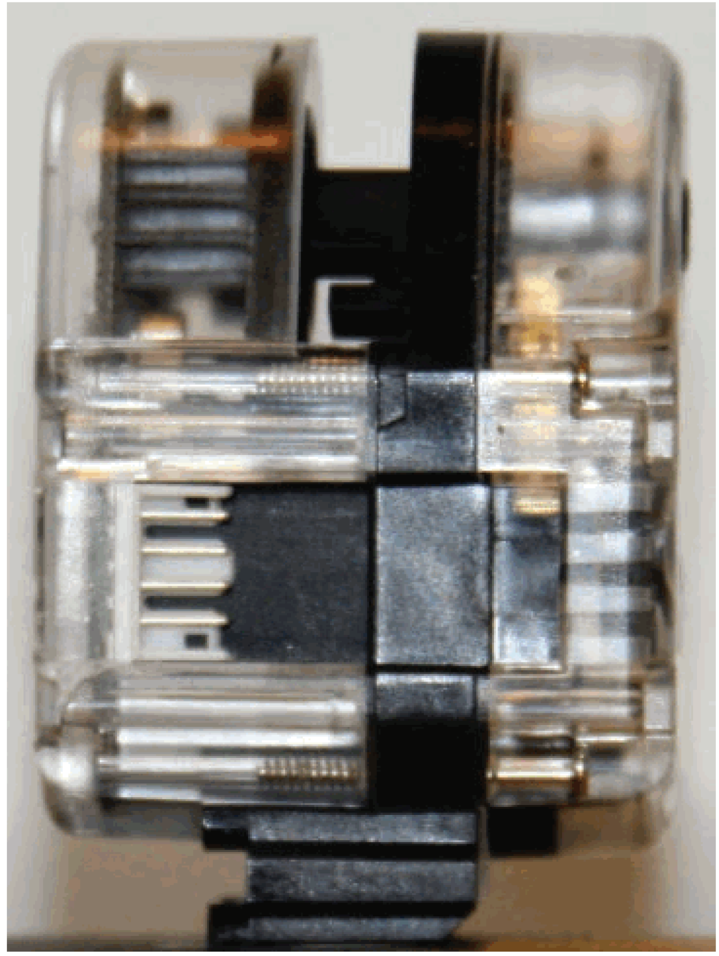


Figure 3B.

Figure 4.


mounting hole options, but plenty to mount the servo in a robot in any way you need. Like the Dynamixel, it has a rear mounting hole in-line with the servo horn axis for a strong joint connection.

The wCK actuator has a completely different mounting scheme and shape to the servo (see **Figure 3**). The wCK does not have a mounting hole on its back side to match the servo horn mount on the front. The wCK actuator has an open slot in the center of the servo that allows a rod to be attached there. This means that the actuator arm is at the center of the servo to begin with.

All three actuator types are of similar size. The AX-12A is a little "beefier" than the DRS-0101, and the wCK units are not square. **Figure 4** shows their relative sizes.

All of these robot servos support a continuous rotation mode and the attachment of wheels. ("Wheels? We don't need no stinking wheels!") These actuators really shine when used to create walking robots.

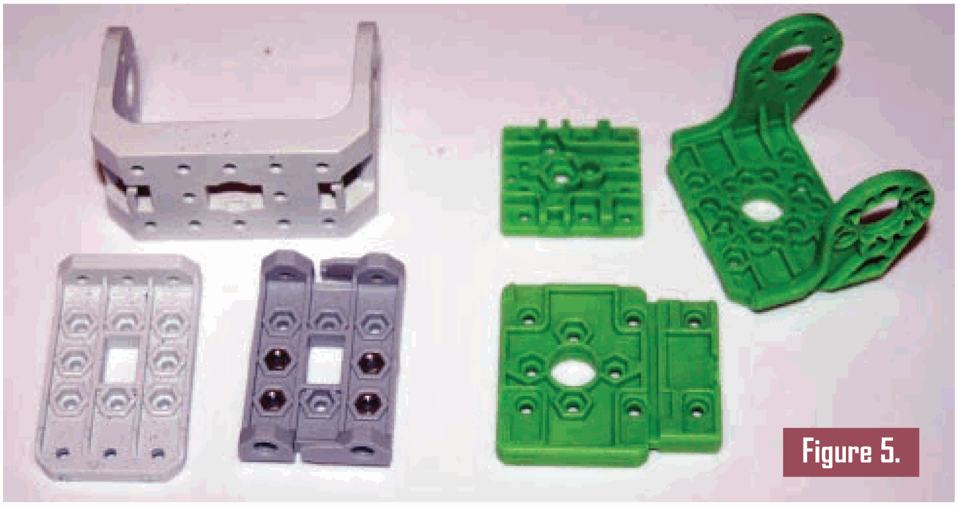


Figure 5.

Toward that end, they all come with various mounting plates and joints for knees, elbows, and feet. **Figure 5** shows some of these options for the Dynamixel and HerkuleX actuators. **Figure 6** shows that the wCK system is radically different.

Whichever system you choose, you can build robust walking robots whose mechanical connections are simple. The one common component to all of these actuators is an enormous number of nuts and bolts.

Well, we've come to the end of another Mr. Roboto. I hope you have learned something that you can use!

Keep those cards and letters coming to roboto@servomagazine.com and I'll do my best to answer them! Until next month, keep those robots running! SV

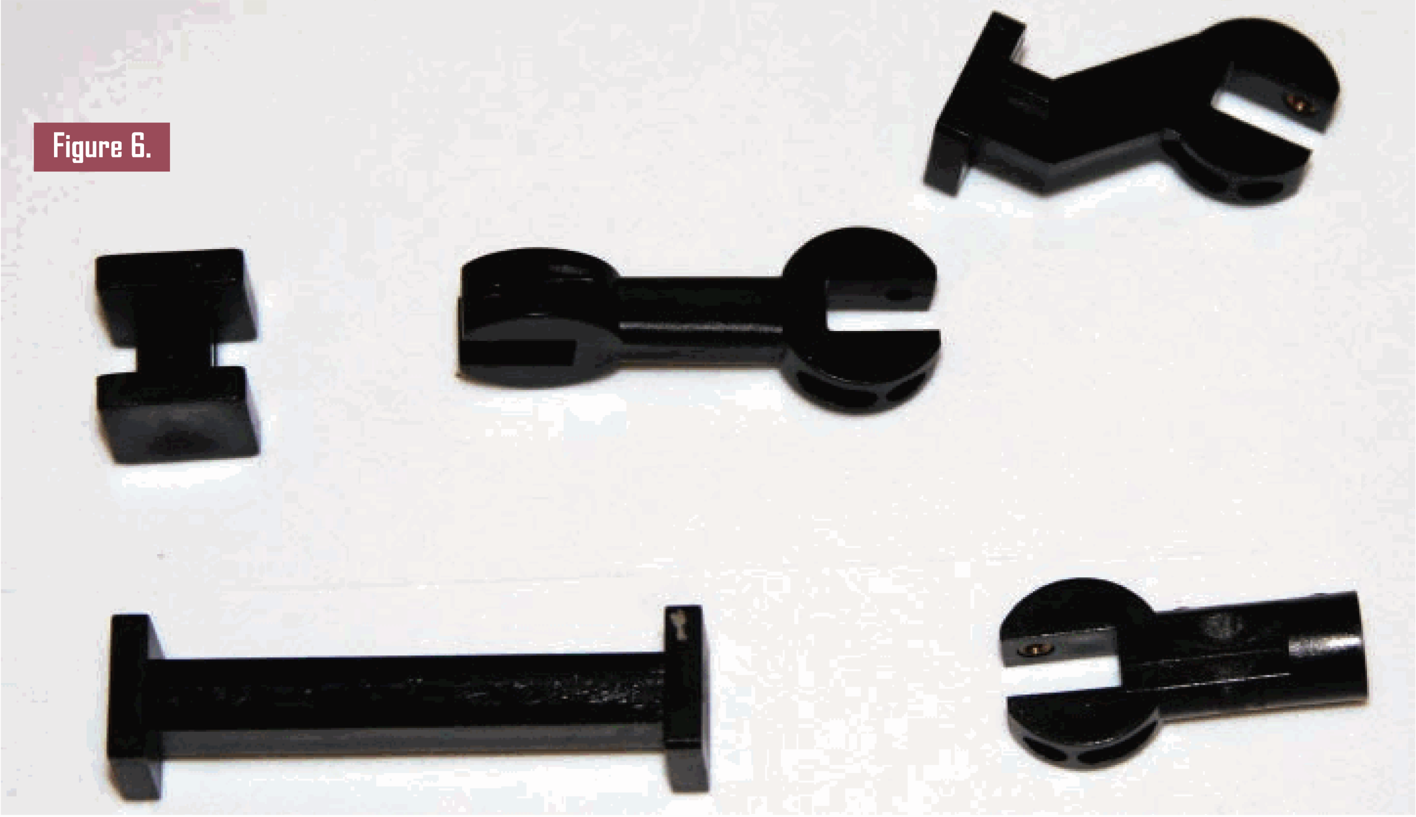


Figure 6.

EVENTS

ROBOTS.NET

Know of any robot competitions I've missed? Is your local school or robot group planning a contest? Send an email to steve@ncc.com and tell me about it. Be sure to include the date and location of your contest. If you have a website with contest info, send along the URL as well, so we can tell everyone else about it.

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**.

— R. Steven Rainwater

OCTOBER

4-5 **CalGames**
Fremont High School
Sunnyvale, CA
A FIRST-based robot event for high school teams.
http://wrrf.org/events/calgames-2013

18-20 **Latin American Robotics Competition**
Fortaleza, Brazil
Events include the Brazilian Robotics Competition, RoboCup Latin American Open, the Brazilian National Olympiad, and the National Robotics Fair.
www.cbrobotica.org

NOVEMBER

2 **Bloomington VEX Tournament**
Bloomington, IN
Events for autonomous and remote-control robots.
https://sites.google.com/site/bloomingtonroboticsclub/

8-9 **Texas BEST Competition**
Curtis Culwell Center
Garland, TX
Remote-control robots built by student teams face off in an annual contest.
www.bestinc.org

9 **STHLM Robotic Championship**
Stockholm, Sweden
Autonomous robots compete in events that include Sumo, folk-race, line following, and freestyle.
www.robotchampion.se

10 **International Micro Robot Maze Contest**
Nagoya University, Japan
One cm cube robots compete in Micro Robot Racer and the Climbing Competition; one inch cube robots compete in Maze Solver; and there's even a two-legged robot event for tiny two inch biped robots.
http://imd.eng.kagawa-u.ac.jp/maze

16 **Atlanta Hobby Robot Club Robot Rally**
Atlanta, GA
See the website for rules and events planned for this year's Robot Rally.
www.botlanta.org

16-17 **Robotex**
Tallinn, Estonia
Events include 3 kg Sumo, iRobot Sumo, LEGO Sumo, Mini Sumo, line following, soccer, and robot racing.
www.robotex.ee

22-24 **All Japan MicroMouse Contest**
Nagareyama City
Chiba, Japan
Classic and half-size autonomous micromouse maze solving plus Robotrace.
www.ntf.or.jp/mouse

24 **Robocon**
Tokyo, Japan
Student teams from over 60 schools compete in the annual robot competition that has been held every year since 1988.
www.official-robocon.com

DECEMBER

16-19 **IROC International Robot Olympiad**
Denver, CO
Events at this year's Olympiad will focus on agricultural robots.
www.iroc.org

ROBOTIS

High-performance networked actuators for robots

DYNAMIXEL

All in One-Actuator

High Power, High Precision

Full Modular Solution

Sophisticated Control Algorithms

Novel Gear Reduction System

Item	Unit	Data			
		H42-20-S300	H54-100-S500	H54-200-S500	L54-60-S300-R
Nominal voltage	V	24	24	24	24
No load speed	RPM	28.3	35.2	35	31
No load current	A	0.61	1.06	1.18	1.1
Continuous speed	RPM	15.59	32.7	32.1	22.4
Continuous torque	Nm	5.596	21.142	39.131	11.8
Continuous current	A	1.989	5.930	9.505	3.1
Resolution	Step/turn	304,000	502,000	502,000	4096
Gear ratio	-	304	502	502	298
Backlash	arcmin	3.5	3.5	3.8	3.5
Interface	-	RS-485 / CAN	RS-485 / CAN	RS-485 / CAN	RS-485 / CAN
Operating temperature	˚C	5~55	5~55	5~55	5~55

DYNAMIXEL PRO

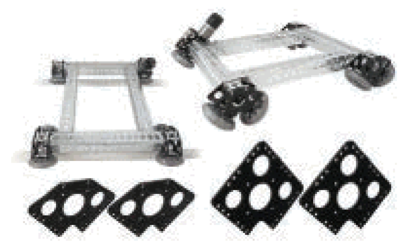
Internal design structure (Left). / Actual product (Right).

<table>
<tr><th>Item</th><th colspan="3">AX-12W</th><th>AX-12A</th><th>AX-18A</th><th colspan="3">MX-28</th><th>RX-24F</th><th>RX-28</th><th colspan="3">MX-64</th><th>RX-64</th><th colspan="3">MX-106</th></tr>
<tr><td>Gear Ratio (material)</td><td colspan="3">32 : 1 (enpla)</td><td>254 : 1 (enpla)</td><td>254 : 1 (enpla+metal)</td><td colspan="3">193 : 1 (metal)</td><td>193 : 1 (metal)</td><td>193 : 1 (metal)</td><td colspan="3">200 : 1 (metal)</td><td>200 : 1 (metal)</td><td colspan="3">225 : 1 (metal)</td></tr>
<tr><td>Network Interface</td><td colspan="3">TTL</td><td>TTL</td><td>TTL</td><td colspan="3">TTL / RS-485</td><td>RS-485</td><td>RS-485</td><td colspan="3">TTL / RS-485</td><td>RS-485</td><td colspan="3">TTL / RS-485</td></tr>
<tr><td>Position Sensor (Resolution)</td><td colspan="3">Potentiometer (300°/1024)</td><td>Potentiometer (300°/1024)</td><td>Potentiometer (300°/1024)</td><td colspan="3">Contactless Absolute Encoder (360°/ 4096)</td><td>Potentiometer (300°/1024)</td><td>Potentiometer (300°/1024)</td><td colspan="3">Contactless Absolute Encoder (360°/ 4096)</td><td>Potentiometer (300°/1024)</td><td colspan="3">Contactless Absolute Encoder (360°/ 4096)</td></tr>
<tr><td>Motor</td><td colspan="3">Cored Motor</td><td>Cored Motor</td><td>Coreless Motor</td><td colspan="3">Maxon Motor</td><td>Coreless Motor</td><td>Maxon Motor</td><td colspan="3">Maxon Motor</td><td>Maxon Motor</td><td colspan="3">Maxon Motor</td></tr>
<tr><td>Operation Voltage (V)</td><td>9.0</td><td>11.1</td><td>12.0</td><td>9.0~12.0</td><td>9.0~12.0</td><td>11.1</td><td>12.0</td><td>14.8</td><td>9.0~12.0</td><td>12.0~18.5</td><td>11.1</td><td>12.0</td><td>14.8</td><td>12.0~18.5</td><td>11.1</td><td>12.0</td><td>14.8</td></tr>
<tr><td>Stall Torque (N.m)</td><td colspan="3">N/A</td><td>1.5 at 12.0V</td><td>1.8 at 12.0V</td><td>2.3</td><td>2.5</td><td>3.1</td><td>2.6 at 12.0V</td><td>2.5 at 14.8V</td><td>5.5</td><td>6.0</td><td>7.3</td><td>4.0 at 14.8V</td><td>8.0</td><td>8.4</td><td>10.0</td></tr>
<tr><td>Stall Current (A)</td><td>1.1</td><td>1.3</td><td>1.4</td><td>1.5</td><td>2.2</td><td>1.3</td><td>1.4</td><td>1.7</td><td>2.4</td><td>1.5</td><td>3.9</td><td>4.1</td><td>5.2</td><td>2.1</td><td>4.8</td><td>5.2</td><td>6.3</td></tr>
<tr><td>No Load Speed (RPM)</td><td>360</td><td>430</td><td>470</td><td>59</td><td>97</td><td>50</td><td>55</td><td>67</td><td>126</td><td>67</td><td>58</td><td>63</td><td>78</td><td>49</td><td>41</td><td>45</td><td>55</td></tr>
</table>

USA E-mail : america@robotis.com / Tel : +1-949-333-3635
JAPAN E-mail : japan@robotis.com / Tel : +81-3-4330-3660
Etc. E-mail : contactus2@robotis.com / Tel : +82-70-8671-2609

www.robotis.com

Dolly Drive & Idler Wheel Plates



ServoCity is now offering their new dolly wheel drive and idler plates which make it easy to assemble a low friction drive system designed to work with 1" OD to 2" OD tubing (and any OD in between). The plates are made from 1/8" Delrin and are designed for use with ServoCity's 2.975" skate wheels. These wheel systems work well with standard PVC pipe (as shown) which can be purchased at a local hardware store. To guide folks through the process of building their own motorized dolly unit, ServoCity includes step-by-step instructional videos on their product page. Motorized dolly systems are great for many applications such as videography, animatronics, robotics, and more. The plates are sold in pairs for $5.99.

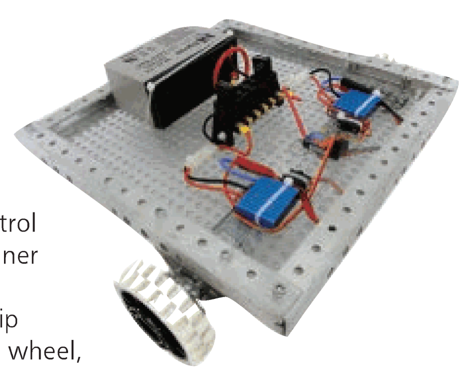
Micro Gripper Kit

ServoCity's new micro gripper kit is a simple solution for projects requiring small pincher-grippers. The kit is easy to assemble and requires only a Phillips screwdriver. Compatible with Hitec's HS-55 servo or HS-5055MG servo, the 6-32 mounting through hole allows for simple attachment to various components, including ServoCity's new Actobotics™ line. The gripper kit includes three 1/8" thick Delrin gripper pieces and the required hardware (servo not included). Price is $6.99 each.

For further information, please contact:

ServoCity	Website: **www.servocity.com**

Affordable Demo Remote Controlled Units



AndyMark, Inc., announces two new separate, affordable, and complete demonstration robot remote controlled units.

Affectionately named the Cheap & Dirty RCU — due to its affordability, compact size, and ease of use — this remote controlled unit is inexpensive and contains quality designed parts. The Cheap & Dirty RCU is a different kind of control system because no programming is needed out of the box, so it is ideal for beginner robotics enthusiasts, hobbyists, and demonstration purposes.

This un-assembled kit includes two pieces of C-channel 35" long, two 4" HiGrip rubber treaded wheels driven by two PG27 gearmotors, a ball caster for the third wheel,

and a complete electronics kit for controlling the two motors. The C-channel can easily make a 15.5" x 15.5" frame. Although assembly instructions on AndyMark's website show one way to build this unit, it can be changed up any way desired, specific to a builder's application.

The controls for the Cheap & Dirty RCU are a helicopter/airplane 2.4 GHz radio control transmitter/receiver combination. List price for this product is $375.

The second demo robot is the ShowCase Portable Trainer. This robot has a small portable design ideal for demonstration and training purposes. This is a completely assembled robot, with the exception of the

four-slot NI cRIO (not included). Mounting holes and hardware are included. Only two wires need to be plugged in for use: the Ethernet and power to the AP.

The electronics board is built to easily flip from storage to demo-ready position, and nearly all the wiring stays in place. At the size of a small carry-on, this unit makes it easy to transport for quick

Continued on page 80

bots

IN BRIEF

GOING APE OVER ROBOSIMIAN

The DARPA Robotics Challenge is mere months away, and now we're getting some early looks at progress on some of the Track A robots. This is RoboSimian — from the Jet Propulsion Laboratory — starting to experiment with hands developed at Stanford.

RoboSimian isn't finished yet, but that's part of what's exciting here. We'll get to see videos of the Track A teams developing and testing their hardware prior to the Challenge in December. JPL is particularly interesting because they've decided not to build a humanoid like most of the rest of the Track A teams. Instead, RoboSimian is more of, well, a simian — a term used most often to refer to apes, although technically we humans are simians too.

In particular, RoboSimian will use its four general-purpose limbs and hands — capable of both mobility and manipulation — to achieve passively stable stances; create multi-point anchored connections to supports such as ladders, railings, and stair treads; and brace itself during forceful manipulation operations.

It looks like RoboSimian is going to have no trouble with ladder climbing or manipulation, and if it ends up walking around on four legs instead of two, that could significantly simplify some of the walking challenges. So the question is, what disadvantages does a form like this have over a more traditional humanoid robot — if any? We may have to wait until the end of the year to find out but in the meantime, keep an eye out for videos from all of the DRC teams.

Here's a video to check out of RoboSimian: **www.youtube.com/watch?v=YsLQ__ycsNU**.

DRONES SAY BUG OFF

Is there anything worse than mosquitoes? Probably, but mosquitoes are pretty bad. Besides being buzzy and itchy and annoying, they can transmit nasty diseases including malaria and West Nile virus, even in civilized (mostly) places like Florida. The issue with mosquitoes is that they're everywhere, and if you've ever tried to get rid of even one mosquito, you can imagine how hard mass eradication is. Well, in Florida, they're about to experiment with aerial drones to see if they can help.

The secret to mosquito eradication seems to be to tackle them in the larval stage when they live in warm shallow pools of water. With chemicals (or hungry fish), you can get rid of the larvae before they take flight. The issue in places like Florida is that warm and shallow pools of water are absolutely everywhere, and finding them all becomes a real problem. The Florida Keys Mosquito Control District will be using a Maveric drone from Condor Aerial equipped with a shortwave infrared camera to see if it's possible to detect pools of water likely to contain mosquito larvae. Once the pools are found, figuratively "nuking" them is relatively straightforward. Here's a video about Maveric drones: **www.youtube.com/watch?v=qkTITZPM6q8**.

ORBOTIX STAYS ON THE BALL

Sphero is a robotic ball that you can drive around with your smartphone. It's a lot fun and it's very cool how you can get down into its software and mess with it, changing it from a relatively simple remote controlled toy into a real autonomous robot. You can even control it with ROS.

Now, Orbotix has introduced the Sphero 2.0 which is packed with new hardware that makes it faster, smarter, faster, brighter, and faster than ever. (Did we mention it's faster? Because it's definitely faster.)

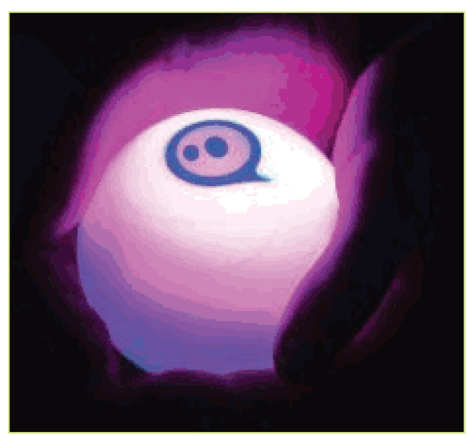
A quick rundown on the Sphero if you're not familiar: It's a roughly softball-sized plastic sphere with a clever system inside that it uses to drive itself around without having to rely on any external motors or controls. Your interface with Sphero is through the Bluetooth connection on your smartphone, and it's got a nifty little induction charging cradle to keep it powered.

The new Sphero (officially called Sphero 2.0) looks much the same as the original one — mainly because it has to be a sphere to work, which limits external design changes. The real changes are inside in both hardware and software. The flagship new feature (as you may have guessed) is that Sphero 2.0 is 2.0x as fast as Sphero 1.0. The original Sphero was lively enough, but this new one is intimidatingly fast, with a maximum speed of well over two meters per second.

Inside, there's a bigger and more powerful motor, fancy new firmware, and the center of gravity has been lowered substantially. Realistically, when you're starting out, this makes the robot nearly uncontrollable which is why when you start with Sphero out of the box, the speed is cranked down a whole bunch.

Unlocking Sphero's top speed involves progressing through a series of driving challenges which sounds annoying, except that you really do need the practice before you're qualified to unleash that full two meters per second. Once you're there, you can send Sphero off of jumps and stuff (included in the box). It actually manages to catch a respectable amount of air.

Sphero's biggest issue when it comes to driving is that by definition it doesn't have a lot of traction, so it has a little trouble accelerating and steering on wood and tile floors. Plus, no matter how talented of a driver you are, you're gonna crash into things. Badly. A lot.



Orbotix now sells a "nubby cover" which is a rubber jacket of sorts that you can tuck Sphero into, giving it additional traction and some shock absorbing capabilities. It also makes it easier to drive Sphero on water, because you can do that since it's waterproof. The other hardware update on Sphero 2.0 is a new set of RGB LEDs that makes it extra glowy. It's three times brighter and you can set it to any color you like using an app.

Potential is what is most awesome about Sphero, however. Because you can record and play back macros, and then look at the code that those macros create and then modify it, Sphero provides a great way to get introduced to simple programming, and with ROS integration you can get as complicated as you want. Just don't forget, it's still a fun little toy, and the 25 apps and games that it comes with should help keep you entertained. Sphero 2.0 costs $130, with 1.0 dropping to $109.

DRONE-ING ON AGAIN

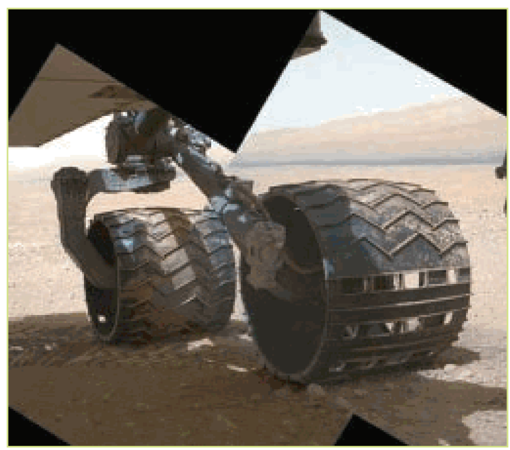
Recently, Northrop Grumman's X-47B unmanned combat air thingy (vehicle or system — take your pick) did a mostly excellent job at autonomously taking off from — and more importantly landing on — an aircraft carrier. Once everything was shown to work, the US Navy was like, "Awesome job, dudes. Now, never fly those things again," and the two X-47Bs were slated for permanent museum display. Fortunately, the Navy has just changed its mind.

The Navy is now planning to deploy the drones to aircraft carriers three more times over the next two years. The first deployment should happen by the end of this year, followed by a second deployment about a year from now, and a final one sometime between late 2014 until early 2015.

From the sound of things, that last deployment is going to be the most exciting one. The X-47B will "fully integrate with a 70-plane carrier air wing for several weeks," to (hopefully) show that robots can seamlessly work with manned aircraft in carrier operations. We'll also get to see the first aerial refueling operation.

In addition to testing out the robotic aircraft more thoroughly, these deployments will also serve to prepare the aircraft carriers themselves for routine drone operations. In many ways, that's the biggest hurdle that the X-47B has to fly over: Getting humans comfortable with having sophisticated and potentially armed robots flying around on their own.

DRIVING MS. CURIOSITY — NOT!



JPL has decided to let Curiosity think for herself, now letting the rover decide where to drive.

By "decide," JPL can now tell Curiosity, "we want you to go over that way and end up at a specific place, but you can figure out for yourself how to get there." To do this, the rover analyzes images from its cameras as its driving to determine which potential routes are safe and which aren't.

"Curiosity takes several sets of stereo pairs of images, and the rover's computer processes that information to map any geometric hazard or rough terrain," said Mark Maimone, rover mobility engineer and rover driver at NASA's Jet Propulsion Laboratory in Pasadena, CA. "The rover considers all the paths it could take to get to the designated endpoint for the drive and chooses the best one."

A recent drive on the mission's 376th Martian day — or "sol" — took Curiosity across a depression where ground surface details had not been visible from the location where the previous drive ended. The drive included about 33 feet (10 meters) of autonomous navigation across hidden ground as part of a day's total drive of about 141 feet (43 meters).

"We could see the area before the dip, and we told the rover where to drive on that part. We could see the ground on the other side, where we designated a point for the rover to end the drive, but Curiosity figured out for herself how to drive the uncharted part in between," said JPL's John Wright, another rover driver.

For the future of robotic exploration of our solar system and beyond, autonomy is going to play a bigger and bigger role, if for no other reason than the time it takes for instructions to get to the robots.

UAV HAS SOLAR FLAIR

The largest robot at the AUVSI expo held recently belonged to Titan Aerospace. It was a model of their Solara 50 robotic atmospheric satellite, and they had to chop off the tail and most of the wings to get it to fit inside the building. The Solara is intended to lift a payload to 20,000 meters and then keep it there for five years, running entirely on solar power. It functions a bit like a satellite, except it is substantially cheaper and much more versatile. Plus, you can get it back when you're done.

It's a little hard to tell, but these UAVs are big. The Solara 50 is 15 meters long with a wingspan of 50 meters, and there's an even larger one called the Solara 60 with a 60 meter wingspan. Despite its size, the Solara 50 only weighs 160 kilograms and it can carry a 30 kg payload, which is fairly respectable.

What makes the Solara functional as an atmospheric satellite are two things. The first is the altitude that it's designed to fly at. At 20,000 meters, you're pretty much above everything. You're looking down on clouds and weather, and the winds and temperatures are generally very stable — or at least predictable. Being that high also gives you a field of view encompassing about 45,000 square kilometers. If you were to mount a cellular base station on a Solara, it would take over for a hundred cell towers on the ground.

The second thing that makes Solara work is that it's solar powered. Every available surface on the wings and tail are covered in solar panels, and there are batteries inside the wings. During the day, Solara generates kilowatts of power, and there's enough left over in the batteries to provide hundreds of watts all night. Because the UAV never requires refueling it can stay aloft for five years, either circling over one spot on the ground, or (if you want it to travel) it's got an effective range of something like 4.5 million kilometers, cruising at just under 60 knots.

The five year life is based on components, so Solara may very well be able to stay up for longer and that's what is so cool about Solara — the fact you can bring it back down if something goes wrong. Even if nothing goes wrong, you still get your payload back at the end of five years — something usually impossible with satellites. Solara is also much much cheaper than a satellite, although the company isn't quite ready to say how much.

THIS WOULD BE MABULOUS

Sometimes it's fun to take a look at impossible concepts, especially if they're: a.) utterly insane; and b.) provide enough foundation for us to convince ourselves that they're not actually completely entirely totally impossible, even if they are.

Adrian Perez Zapata's futuristic concept — which he calls "Mab" — envisions a swarm of tiny flying robots zipping around your house to clean surfaces before returning to a spherical home base.

Mab is a self-cleaning system consisting of 908 robots which clean the surface of a floor by touching and trapping the dirt particles on the floor with droplets. These robots also fulfill the task of feeding the system energy by capturing solar energy in their wings. The second component of the Mab is the core which the robots returns to. This central part handles multiple tasks: It generates the mixture of water with an additive that gives higher surface tension and a pleasant odor to the water; it controls the robots based on information they are providing about the environment; it receives the contaminated droplets and filters them to remove the dirt from the water, saving the highest percentage possible; and cleans its walking surfaces.

The following summarizes the seven step cleaning process:

1. Mixes the water and the substance that gives greater surface tension.
2. The mixture is distributed to subordinates (robots).
3. The robots fly with the load. The robots use a propeller for flying.
4. The robots clean by touching the surface with droplets of fluid.
5. The droplets capture the dirt and carry it back to the core.
6. The core filters the dirt out.
7. The core recovers the highest possible percentage of water to restart the cycle.

The thought behind Mab is to restore a sense of wonder in everyday life, and to recapture the magic in simple processes, providing human shelters an autonomous purification.

It's a little bit hard to tell from the pictures just how small the flying robots are, but they're seriously tiny. Getting robots that small to fly at all — much less fly intelligently — is exceptionally difficult but not impossible, as Harvard is trying to show with their Robobee project at **http://robobees.seas.harvard.edu**/.

BACK WITH NEW BLACK

Parrot's latest AR drone upgrade — the Power Edition — features a new piano black hull, up to 36 minutes of flight time, and some swanky new prop colors.

Keep in mind that the "36 minutes of flight time" is the total flight time you get using both the high capacity batteries (if you land the drone and switch a spent battery for a fresh one). The drone does not carry both batteries at once and besides the color, there is no fundamental difference in the hardware. The batteries will run you about 60 bucks by themselves, so the whole package (including four sets of props) costs about $370 — not too bad a deal.

MRI-GUIDED BOT TO SEARCH AND DESTROY BRAIN TUMORS

With a five-year $3 million R01 award from the National Institutes of Health (NIH) through the National Cancer Institute (NCI), a team of researchers led by Gregory Fischer, PhD — assistant professor of mechanical engineering and robotics engineering at Worcester Polytechnic Institute (WPI) and director of WPI's Automation and Interventional Medicine (AIM) Laboratory (**aimlab.wpi.edu**) — will test a new minimally invasive approach to treating brain tumors that promises to accurately destroy malignant tissue while leaving surrounding tissue unaffected. This approach would be a significant improvement over current treatments.

The system will use a robot designed to work within the bore of an MRI (magnetic resonance imaging) scanner to precisely guide a probe through a dime-sized opening in the cranium to the tumor with the aid of real time MRI images. The probe will destroy the tumor by heating it with interstitial high-intensity focused ultrasound (iHIFU).

Developed by industry collaborator Acoustics MedSystems, Inc., the device can emit ultrasound energy in a highly directional manner so only malignant tissue is heated — even with irregularly shaped deep brain tumors. When guided by live MRI images — using a novel robotic manipulator developed by Fischer's lab and specially designed MRI coils developed by Reinhold Ludwig, PhD, professor of electrical and computer engineering at WPI — the probe will be able to accurately target the tumor.

Currently, patients diagnosed with brain tumors typically face one of two courses of treatment, both of which have important limitations. Stereotactic radiation surgery — in which a radiation beam is focused on the tumor — is noninvasive and can increase survival, but it may take multiple treatments to relieve symptoms and it is difficult to confirm that the tumor has been destroyed. Open-brain surgery provides quick relief of symptoms and tissue samples for lab testing, but it is highly invasive and can lead to serious complications.

Acoustic MedSystems will develop the MRI-compatible ablation device and software to help guide and control it. The device will have tiny sensors that will enable doctors to precisely track its position in real time MRI images and an array of ultrasound emitters that will permit the zone of penetration of ultrasound to be adjusted to match the shape of the tumor as it appears in the live images.

The team at UMass Medical School will bring their expertise in MRI imagery to the research and will also coordinate and conduct clinical tests of the robotic ablation system. Fischer's team will develop a new robotic device specifically designed to manipulate and deliver the ablation tool to the proper location in the brain under live MRI guidance.

TOO SEXY FOR THE CATWALK

The annual Robot Fashion Contest in Osaka, Japan is one robot competition that depends more on style and beauty than on technical merit or raw power. "Performance" takes on a totally different meaning.

Competitors are judged on esthetic factors, and the rules are fairly free and open. While there are the usual restrictions — like no extreme weapons, gunpowder, gasoline, and the like — the robots can use props, background music, and even assistants (human or otherwise). Most of the robots are typically humanoid, but the rules allow you to enter other types of robot designs.

Prizes are awarded, and some top competitors will walk away with over USD$1,000. The contest is slated for November 24th.

COMBAT ZONE

Featured This Month:

Last month, Zach Witeof was listed as the writer of the Product Review on the Botbitz 30A ESC. However, it was actually Pete Smith who wrote it. I blew it and apologize to Pete. Kevin Berry

Go to **www.servomagazine.com/index.php?/magazine/article/october2013_CombatZone** for any additional files and/or downloads associated with this article. You can also post comments.

BUILD REPORT:

The Quest for a Different Kind of Flipper: Building the Beast

● by Zac O'Donnell

In a recent issue of *SERVO*, I wrote about scaling up my successful six pound Mantisweight class flipper Threecoil into a 30 pound Sportsman class robot. The initial designs that emulated the flipper geometry ended up being too heavy, so I switched to a different style of flipper built on top of the same flywheel, clutch, and trigger mechanism concept that had proven itself in Threecoil. Once the design of the robot was finalized (**Figure 1**), I started building part orders and a plan to create the machine.

The first shipment contained materials for the baseplate of the robot, so I wasted no time in printing out the design and taping it down to the two foot square aluminum sheet (**Figure 2**). After I finished the baseplate, I moved on to a test-fit of the unfinished weapon rails, flipper head, and flywheel that were made by a local shop (**Figure 3**).

I decided to follow that up with the other pieces of internal framing and mounts (**Figure 4**).

Once the simple parts were done, I moved on to the finer details of the flipper assembly. I started by finishing the coil

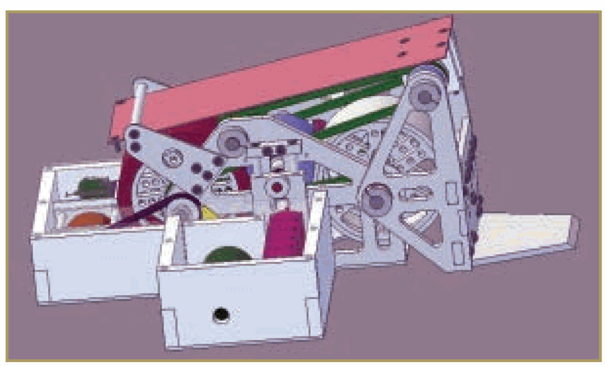


Figure 1. Finished design.

drum end caps and then assembling the coil drum (**Figure 5**). Once the coil drum was finished, I cut the titanium shafting down to size and did a test-fit without bearings for all of the components (**Figure 6**).

While I waited for the bearings to arrive, I started working on the parts for the clutch carriage as seen in **Figure 7**. The bearings finally arrived, so I finished off the flipper assembly by mounting the trigger servos and adding the return springs to the clutch (**Figure 8**).

I added a very thin rope to the assembly to work as a shear pin for my first test, and then installed the whole thing on the baseplate with the flywheel and drive motors (**Figure 9**).

The batteries that I wanted were out of stock, so I ended up making the five-cell battery packs myself. You can see three of the four completed packs in **Figure 10**.

The next logical step after the batteries was the electronics box at the back of the robot and the wiring between all of the different components. This was a bit tricky because of all the moving parts in the flipper assembly, but you can see the results of several different attempts in **Figure 11**.

Before I could do any tests of the entire robot, I had to finish up the frame (**Figure 12**). Once that was done, I cut and attached extra

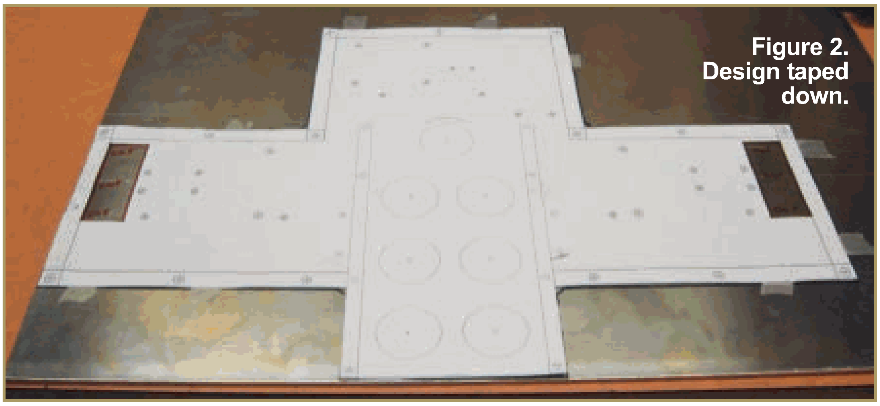


Figure 2. Design taped down.

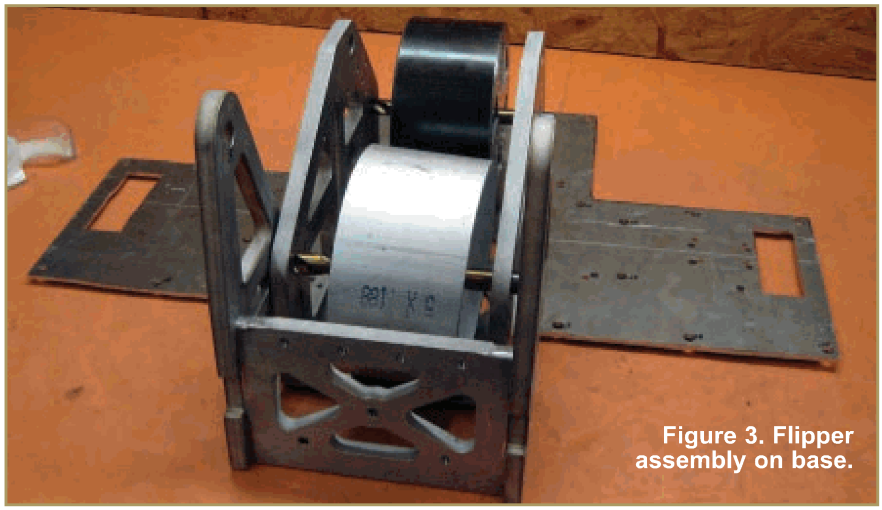


Figure 3. Flipper assembly on base.


Figure 4. Internal mounts.

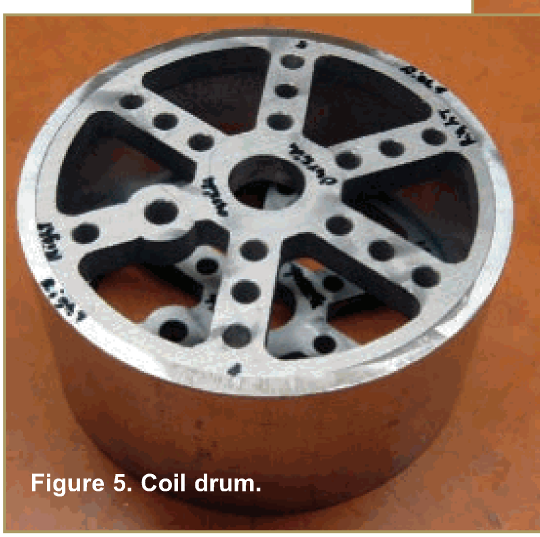


Figure 5. Coil drum.

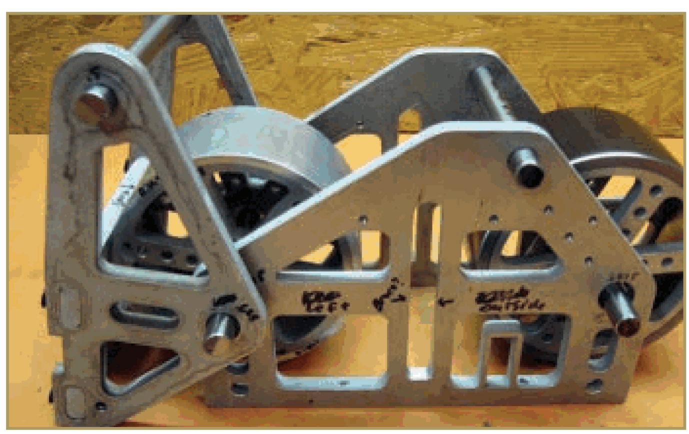


Figure 6. Shafting test-fit.

armor panels and top plates to protect the electronics.

I did extensive testing and ended up making a few changes, but after almost 200 hours of work I dubbed the robot Jack Reacherafter in honor of my favorite fictional character. I added a little paint and you can see the final result in **Figure 13**. **SV**

Figure 10. Battery packs.

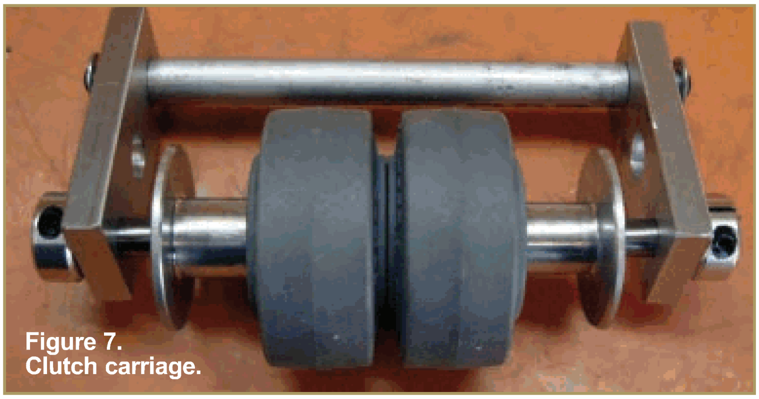


Figure 7. Clutch carriage.

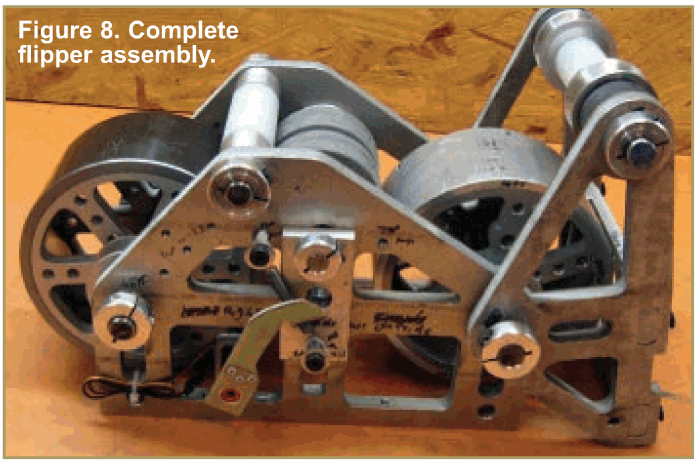


Figure 8. Complete flipper assembly.

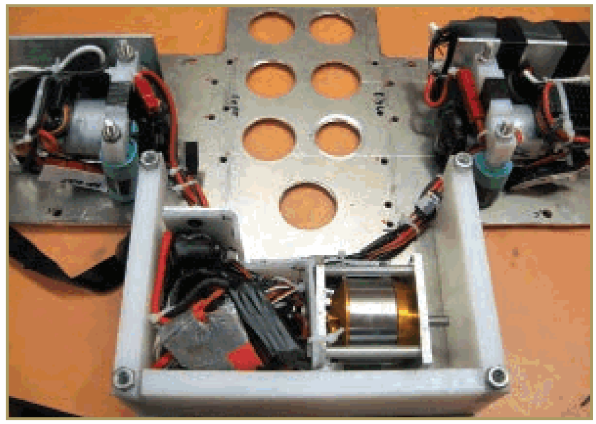


Figure 11. Wiring harness.

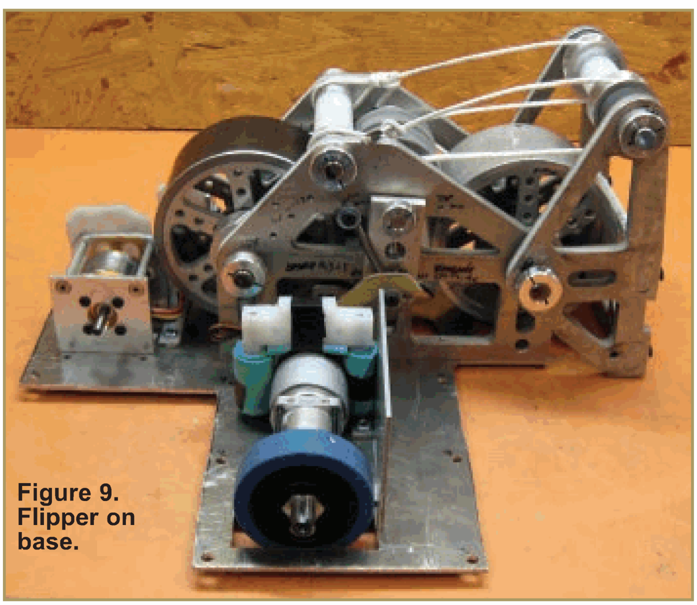


Figure 9. Flipper on base.

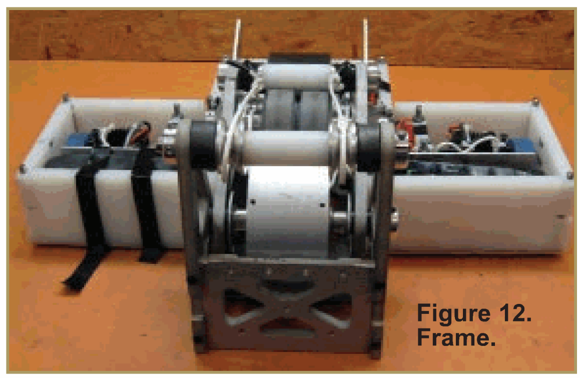


Figure 12. Frame.

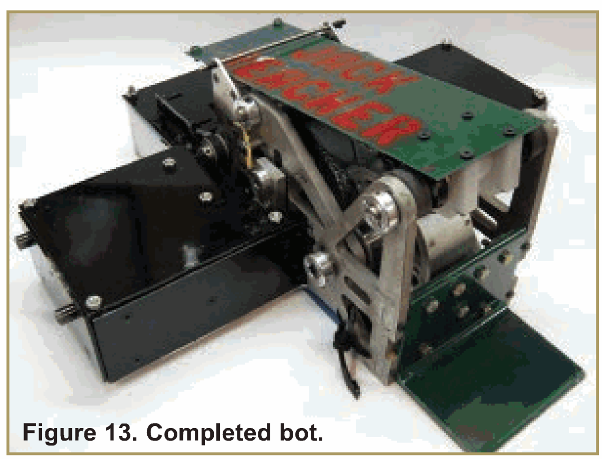


Figure 13. Completed bot.

National Robotics League Begins Their 6th Year of Combat Competition

● by Kevin M. Berry

The National Robotics League (NRL) 2013 Championships held in Indianapolis this last May, featured 48 robots built by nearly 200 students from 25 schools across the US. The grueling, exciting, and educational weekend faceoff of student-built robot gladiators wrapped up with the robot "Pandemonium" from Eastern Westmoreland CTC, Latrobe, PA, taking the title of Grand Champion. Created and sponsored by the National Tooling and Machining Association (NTMA), the NRL is designed to help introduce a new generation of students to the advanced skills and technology of today's manufacturing. The competition has doubled in size in the past two years alone, with twice the number of students participating and twice the number of robots entering the competition.

The NTMA was first introduced to the idea of educational combat robotics via a BotsIQ event in 2005. When machine shop owners from across the country saw how involved the competitors were in the event, the skills they were learning, and the machines they built, they knew the NTMA needed a program like this to help connect the manufacturing industry with the next generation of technology leaders.

The NTMA began working with BotsIQ in that same year, and then in 2008 decided that in order to really grow the program in NTMA chapters and emphasize the program's connection to the manufacturing industry, a separate program was needed. Thus, the National Robotics League was born.

While the robots built by NRL teams obviously share a lineage with BattleBots, BotsIQ, and all the combat robotics programs to come before it, the NRL is unique in that it places a large emphasis on connecting teams to industry partners — not only for financial and material contributions, but also to serve as mentors and role models who can help students see how the work they are doing on their robot translates directly into experiences and skills that are highly sought by manufacturing employers and engineering school recruiters.

Managed by the NTMA, the NRL was founded thanks to a grant from the National Tooling and Machining Foundation — a 501(c)3 non-profit designed to fund manufacturing education. Begun 20 years ago and run independently from the NTMA, the Foundation provided the seed funding for the NRL because combat robotics is a nice fit for their mission which is to help members of the US precision custom manufacturing industry achieve profitable growth and business success in a global economy through advocacy, advice, education, networking, information, programs, and services.

The event format is similar to that of BotsIQ in that a 15 lb weight limit is imposed. True walkers are awarded a five pound weight allowance. Bots can't be larger than

3′ x 3′, and multi-bots are allowed. Most forms of batteries are allowed except for LiPoly. Pneumatics can use 150 PSI compressed air, or single-use CO_2 cartridges. Only spread spectrum R/C is allowed. Three minute fights are held in a 12′ x 12′ arena. If there are enough entries, the NRL classifies bots by middle school, high school, and post secondary builders.

Given the educational nature of the event, it's no surprise that build documentation is required, and students participate in team interviews with judges. More information is available at **www.gonrl.org**. **SV**

<u>2013-2014 NRL Events</u>

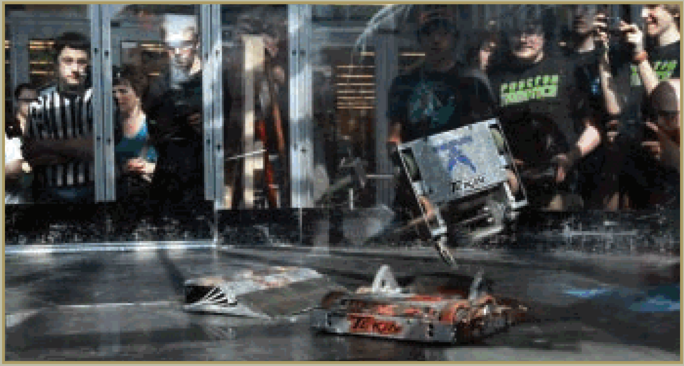
Regional Competitions

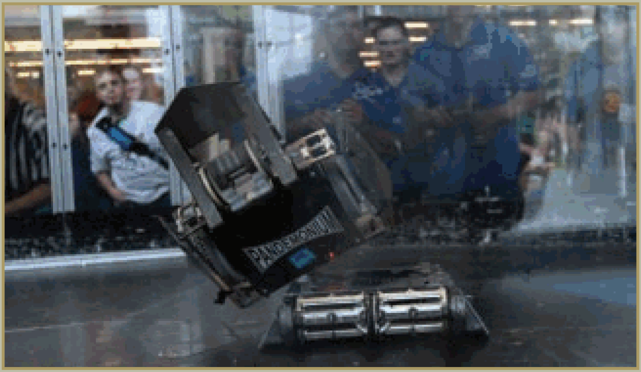
- November: Milwaukee, WI; Minneapolis, MN; and Dayton, OH
- December: Plymouth, MA and Minneapolis, MN
- January: Minneapolis, MN
- February: Minneapolis, MN
- March: Southwest PA; Southeast PA; Southern CA; Dayton, OH; Minneapolis, MN
- April: Dallas, TX (tentative); Boston, MA; Minneapolis, MN; Southwest PA; Cleveland, OH; Bloomsburg, PA; Utah (tentative); and Mesa, AZ (tentative)

May: 2014 National Competition

<u>2013 NRL Championship Winners</u>

- Grand Champion: Pandemonium from Eastern Westmoreland CTC, Latrobe, PA
- 1st Place: UC3 from North High School, St. Paul, MN
- 2nd Place: Pandemonium from Eastern Westmoreland CTC, Latrobe, PA
- 3rd Place: Ramses from Beaumont School, Cleveland Heights, OH
- Best Documentation: Pandemonium from Eastern Westmoreland CTC, Latrobe, PA
- Best Engineered: UC3 from North High School, St. Paul, MN
- Coolest Bot: Jack in the Box from Ponitz Career Center, Dayton, OH





EVENT REPORT:

Robot HORDs Return to The Gate

● by Chris Olin

The Ohio Robotics Club (ORC) in association with the Robot Fighting League and Northern Ohio Radio Control Auto Races (NORCAR) hosted the House of Robotic Destruction 2013 (HORD) at The Gate — North East Ohio's premier indoor R/C car race track — this past May in Burnswick, OH. Eleven teams brought a total of five 150 gram Fleaweights, 11 one pound Antweights, and six three pound Beetleweights to compete in three brutal double elimination tournaments.

The father-son duo of Glenn and Warren Puvin dominated the Flea and Ant brackets with their three hard hitting robots. Their Fleaweight Demise used its front mounted spinning disc to cut a path straight through the winner's bracket to first place.

Meanwhile, the gruesome twosome of Low Blow and Vile Ant scored win after win with their under cutter bars of spinning death. Low Blow took first place with no losses, and Vile Ant finished second — his only defeat being a forfeit to Low Blow.

Next, the pair took on all comers in a massive Antweight rumble. The mighty spinners could just not hold out against the six pushy bots arrayed against them. Both fell to the pits and at the end of the rumble only Don Jenkin's Bully was left standing.

In other action, Richard Kelley's Fleaweight dust pan/clamp bot CLAW arm wrestled Chris Olin's servo-powered lifter Left twice — first losing to Left by being flipped on his side and then in the loser's bracket driving Lefty into the pit. CLAW went on to face and defeat Zach Witeof's Chairman Meow before losing to Demise in the final.

HORD veteran Witeof was joined this year by his fiancée, Alyssa Dole, who drove a new bot Wedge TASTIC!, featuring 20% more wedge-ness than the leading competitor. Her first and only victory was against Bully in the loser's bracket. Alyssa shared the Rookie of the Year award with fellow newcomer and friend of Zach, Steve Nagar whose Boxy Brawler also went 1-2.

Zach recently earned a Master's Degree in Electrical Engineering from Ohio State and plans to continue his studies in Florida. All ORC members and competitors offer Zach and Alyssa their best wishes.

Meanwhile, the Beetle brackets presented a full on pushing war featuring five fast and furious wedge/brick bots and a lone lifter

Builders: (1) Don Jenkins; (2) Chris Olin; (3) Zach Witeof; (4) Steve Nagar; (5) Ron Baron; (6) Warren Puvin; (7) Richard Kelley; (8) David Graham; (9) Chad Savage; (10) Alyssa Dole; and (11) Glenn Puvin.

Meerkat Mreow disassembled by Vile Ant.

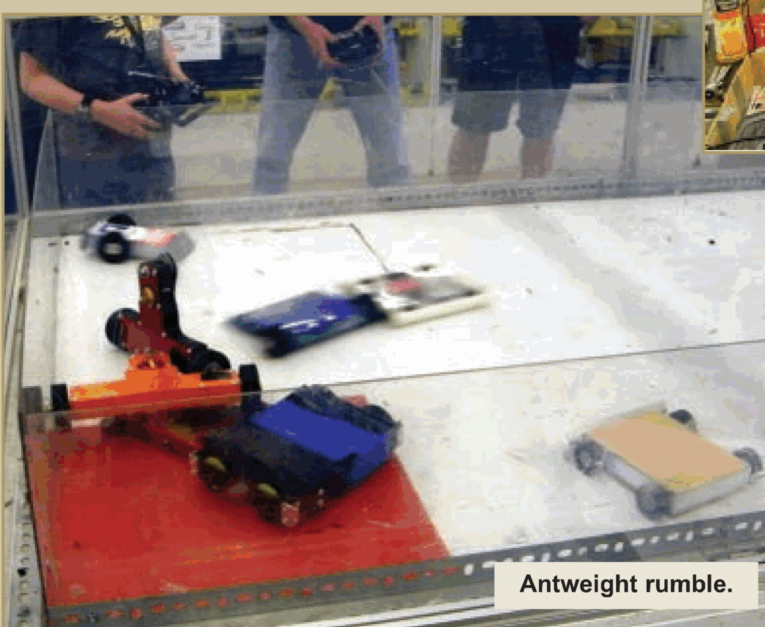
Alyssa Dole driving Wedge TASTIC!

Don Jenkins driving Bully.

bot bent set on revenge. The perennial whipping boy of the Beetle brackets Sweaver (driven by ORC cofounder Ron Baron) pushed through the winner's brackets beating Zach Witeof's wedge bot Kerfuffle, and Chris Olin's servo lifter Revenge of Dr. Super Brain. He then faced Don Jenkins' MoveIt. In the end, Sweaver's wedge beat out MoveIt's bulldozer blade, giving the championship to Sweaver. Dr. Super Brain settled for third.

ORC would like to thank this year's volunteers Tiffany Clegg and John Olin from their help, and this year's sponsors *SERVO Magazine*, **ServoCity.com**, FingerTech Robotics, and Dimension Engineering for their support. ORC is also grateful to NORRAC for allowing them the use of their wonderful facility. **SV**

In the pits, Zach Witeof (left back) and Richard Kelley (right near).



Antweight rumble.

Beetle Rumble — Revenge of Dr. Super Brain defies the pits.

	Bot Name
1	Baby V
2	MoveIt
3	Revenge of Dr. Super Brain
4	99
5	Wedge TASTIC!
6	Hacker
7	CLAW
8	Lefty
9	Sweaver
10	Purple Boxer
11	Vile Ant
12	Don't Get Pushy
13	Bully
14	Demise
15	Boxy Brawler
16	The Froogin
17	Chairman Meow
18	Sidewinder
19	Mateo
20	Low Blow
21	Meerkat Mreow
22	Kerfuffle

TABLE 1 - FLEAWEIGHT WINNERS.

Rank	Bot Name	Team Name	Driver's Name	Team City / State
1	Demise	Pretzel Robotics	Warren Purvin	Harrison Twp., MI
2	CLAW	Kelley PA	Richard Kelley	Boiling Springs, PA
3	Chairman Meow	Trojan Engineering	Zach Witeof	Columbus, OH
4	Lefty	Cloak & Dagger Robotics	Chris Olin	Aurora, OH
5	Baby V	Team Mateo	David Graham	Gettysburg, PA

TABLE 2 - ANTWEIGHT WINNERS.

Rank	Bot Name	Team Name	Driver's Name	Team City / State	Special Awards
1	Low Blow	Pretzel Robotics	Warren Purvin	Harrison Twp., MI	
2	Vile Ant	Pretzel Robotics	Warren Purvin	Harrison Twp., MI	
3	Don't Get Pushy	Beastly Bots	Chad Savage	Rootstown, OH	Most Improved Driver
4	The Froogin	FishNecks	Ron Baron	Garrettsville, OH	
5	99	FishNecks	Ron Baron	Garrettsville, OH	
5	Mateo	Team Mateo	David Graham	Gettysburg, PA	
7	Boxy Brawler	Tractable Engineering	Steve Nagar	Columbus, OH	Rookie of the Year
7	Wedge TASTIC!	Trojan Engineering	Alyssa Dole	Columbus, OH	Rookie of the Year
9	Hacker	Kelley PA	Richard Kelley	Boiling Springs, PA	
9	Bully	Team Labyrinth	Don Jenkins	Parma, OH	Rumble
9	Meerkat Mreow	Trojan Engineering	Zach Witeof	Columbus, OH	

TABLE 3 - BEETLEWEIGHT WINNERS.

Rank	Bot Name	Team Name	Driver's Name	Team City / State	Special Awards
1	Sweaver	FishNecks	Ron Baron	Garrettsville, OH	
2	MoveIt	Team Labyrinth	Don Jenkins	Parma, OH	
3	Revenge of Dr. Super Brain	Cloak & Dagger Robotics	Chris Olin	Aurora, OH	
4	Purple Boxer	Kelley PA	Richard Kelley	Boiling Springs, PA	
5	Kerfuffle	Trojan Engineering	Zach Witeof	Columbus, OH	Rumble
5	Sidewinder	Team Mateo	David Graham	Gettysburg, PA	

Robot Combat Gets Savage at HORD 2013

● by Tiffany E. Clegg

Chad with Beetleweight robot Rippy at HORD 2011.

Young up and comer, 12 year old Chad Savage from Edinburg Twp., OH took third place in the Ant class at the 2013 HORD (House of Robotic Destruction) tournament held in Brunswick, OH.

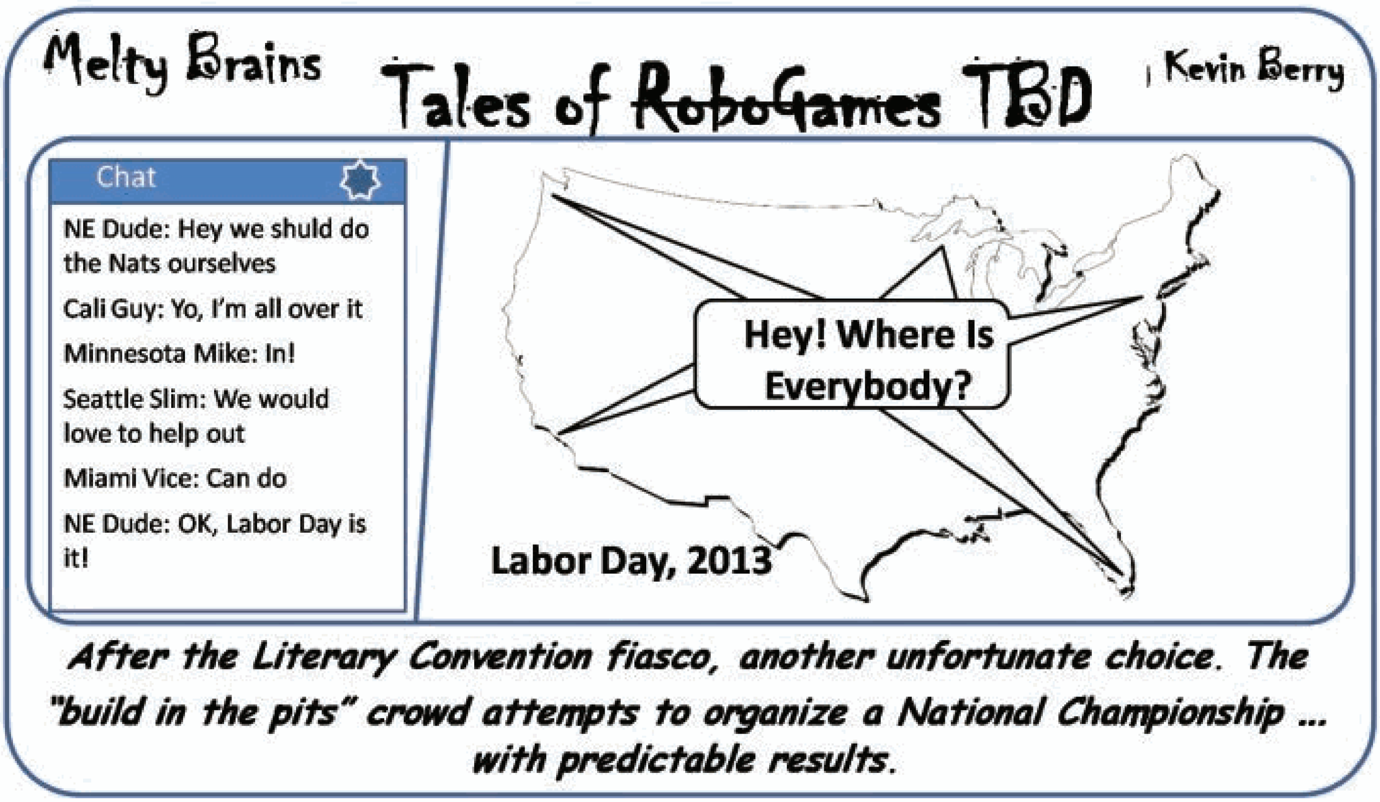
Chad made his way into the top three by defeating several older, more experienced builders with his pusher robot Don't Get Pushy. The boxy, four-wheel drive robot was constructed out of carbon fiber with aluminum angle iron to reinforce its corners.

Chad's interest in robotics developed around the age of nine. By the time he was 10, he had started competing. One of his first competitions was in the elementary division of the National Robotics Competition held in Marion, OH where he took first place in the maze solver event.

The maze bot had to find its own way through the maze based on the programming and sensors which told it when and where to turn. According to the judges, Chad's robot was unique being the first one that they had seen in competition with rear steering.

The following year, Chad again competed and took first place at the Nationals. This time, he entered the rescue robot competition. The course was designed with challenging terrain which included mountains and caves to simulate a wilderness rescue operation.

While navigating the difficult course, the robot had to retrieve a series of balls representing stranded



humans from pylons, taking them through the course to safety. It was after these competitions that Chad decided to keep building because he "... had so much fun."

Since then, Chad has competed in several competitions. Chad — who loves to learn — believes that being the youngest competitor has given him an advantage that others may not be so fortunate to have. Chad recalls the first year he came to HORD, "My robot didn't do so well. Mr. [David] Graham let me drive one of his robots and showed me a lot of stuff. I like to listen and talk to the other people so that I can learn from their experience."

Chad's love of learning and passion for robotics has become a family affair. "We build them together and it's something my whole family can work on. Dad helps me with a lot of the stuff and testing. We have time set aside that we work each week. He enjoys the time we spend together doing it."

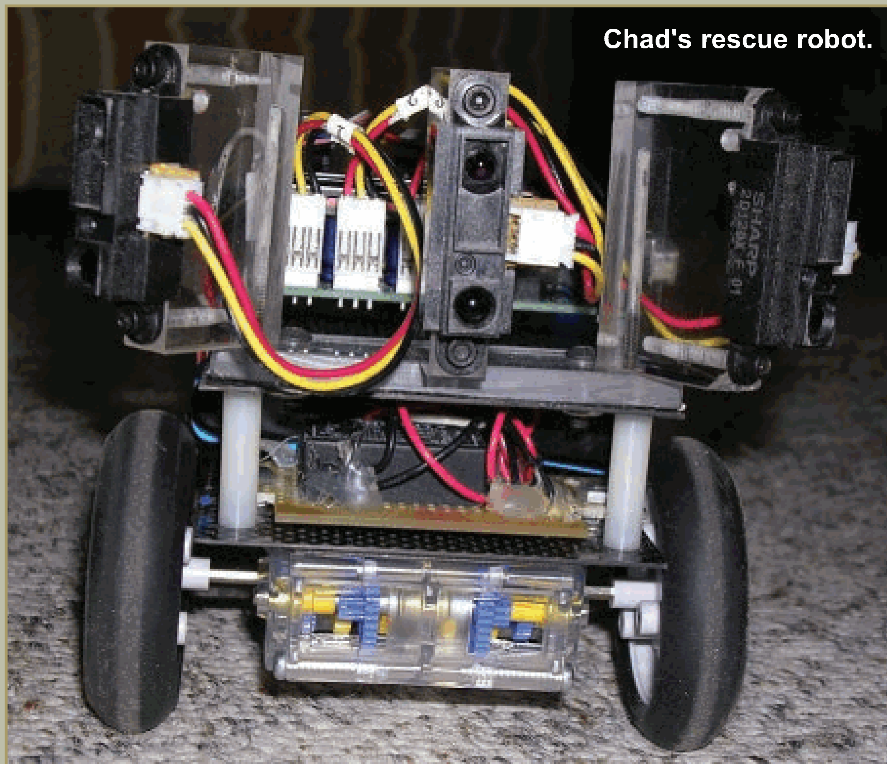
Robotics has also helped Chad to develop a supportive and growing fan base. According to his mother, "We have friends who have helped us with learning and finding access to design systems; given us access to their machinery to design parts; have built parts for us; helped us with scrap material and parts; helped Chad with ideas and set up other learning experiences for him. We even have one who builds us duplicates of the play fields for some of his competitions like the huge maze and gravel pit that sits in our basement."

"It's not just us that encourages him. He has way more than your average support group, and we are extremely grateful to all of them for being a part of his life. The rides home are always filled with phone calls and texts to many people about how the competition went. One of our friends even makes coconut cream pies and cookies to celebrate the wins.

They all have different talents and have been willing to share them with Chad. His dad works for Precision Machine Controls, and they have also been very supportive of Chad and helped with various things including some of our manufacturing and machining of parts."

Chad's love for robotics has so inspired his parents that they are currently raising funds and looking for space to start a children's robotics group in their community.

Chad encourages kids who think they might be interested in robotics to ".... do some research, take chances, and explore your options. Unless you try it, you won't know if you like it." SV


Chad's rescue robot.

Chad receiving award at HORD 2013.

RoboGames

From an International Perspective

One of the most interesting and enlightening aspects of RoboGames is the "international" aspect. Robotics truly has global appeal. International teams come to compete in a wide variety of events, and share their experiences with other robotics enthusiasts from around the world. Entrants from 17 countries competed at the 2013 event, with many international teams taking home Gold, Silver, and Bronze medals. One such group was a team from the school, Lycée Colbert from Lorient, Brittany, France.

By Ray Billings

Go to **www.servomagazine.com/index.php?/magazine/article/october2013_Billings** to comment on this article.

Team Lycée Colbert, enjoying the sites of the San Francisco Bay area.

Lycée Colbert provides a three year course of further secondary education for children between the ages of 15 and 18. This would roughly be the equivalent to a private high school (with college level studies) here in the states. As such, there are many required courses of study for all the students and some options for elective classes. One of the options is the Robotics course which led to their involvement in this year's RoboGames event.

The purpose of the robotics course is to make students discover electronics and mechanics by building a robot. The focus of training is very much "hands on" with students gaining a variety of skills in virtually all aspects of the design and construction of a functional robot. The robotics instruction is broken up into several modules, including electronics, mechanics, and English studies.

The aim of the electronics portion is to help students understand basic electronics, and specifically what is used on the circuit boards controlling the robots. So, for instance, they learn:

- How to make a counter using a seven-segment display.
- How to control the lighting of a power LED from a microcontroller (a PIC16F88) using a transistor.
- How to control the lighting of an RGB LED strip

ribbon from a microcontroller using three transistors.
- How to control a servomotor from a microcontroller by building a PWM.
- How to control a small speaker from a microcontroller.
- How to check a battery's state of charge using potentiometers and operational amplifiers.

The students have to test the different circuits on a prototyping board. The microcontroller is programmed using the assembly language of the PIC16F88. It's a low-level programming language, but it is suitable for the simple program they have to write. Finally, they have to design the circuit board, solder in the components, and test functionality.

For the mechanics portion of study, many different robot types are chosen for consideration. Students have to create presentations of specific robots that interest them and explain how they function to the other students. Based on the presentations of actual robots, the students make suggestions on potential designs to build. Students vote for the robot of the year, and once a victor is chosen the design of the mechanical parts of the robot can begin.

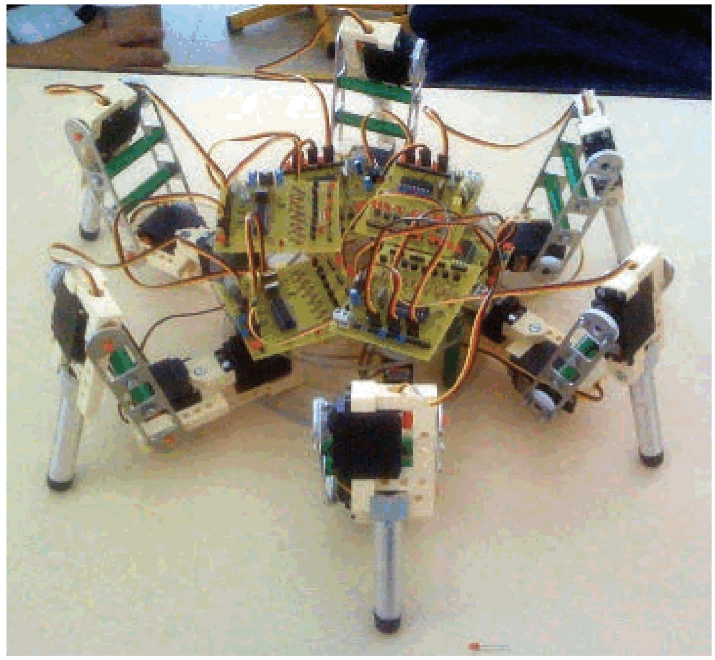
To design the robot's physical components, the students use SolidWorks. All the parts are manufactured in the workshop at the school by the students. Finally, again, they have to assemble the robot and test functionality.

In order to improve their English proficiencies, the entire course is taught in English. However, some activities are more relevant than others to English proficiency. For example:

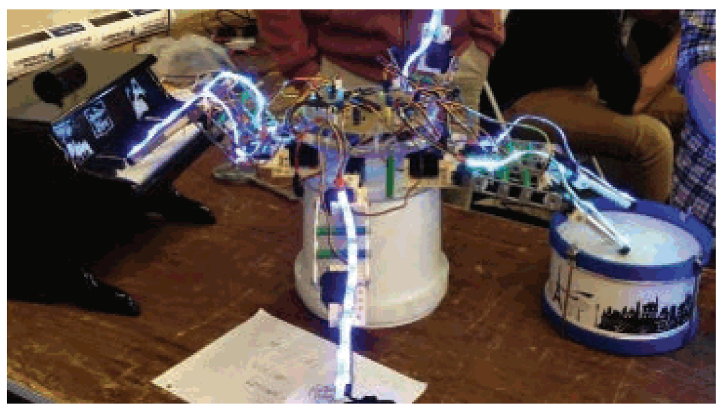
- Much of the presentation work is in English.
- Historical events such as Turing's life are studied.
- Videos about the RoboGames competition are watched.
- The rules of RoboGames are studied.
- Their visit to San Francisco for the event is planned.
- Articles for the blog of the class are written.

The team's entrants this year included Gypsy (hexapod walker), Simulo (four-legged walker), and Nony (bipedal walker). For this event, the students had to run demonstrations to show their robots and explain how they work. This was no easy task for these high school French students to give technical demonstrations in English!

People were very impressed with both the robots and the students. Simulo and Nony were entered in the junior division Best of Show, winning the gold medal and bronze medal, respectively. This year, Gypsy was converted from a walking robot to a musical robot, taking the gold medal in the Art Bot-Musical category.

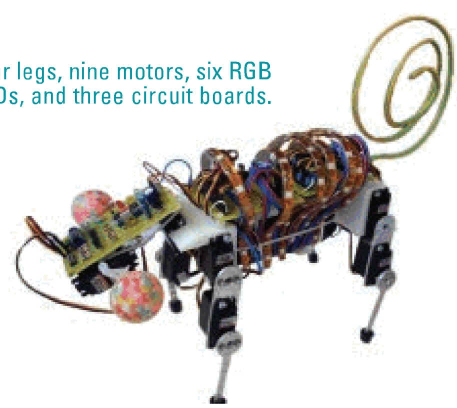


Gypsy was the first robot built for this program. six legs, 18 motors, 36 LEDs, and six circuit boards!

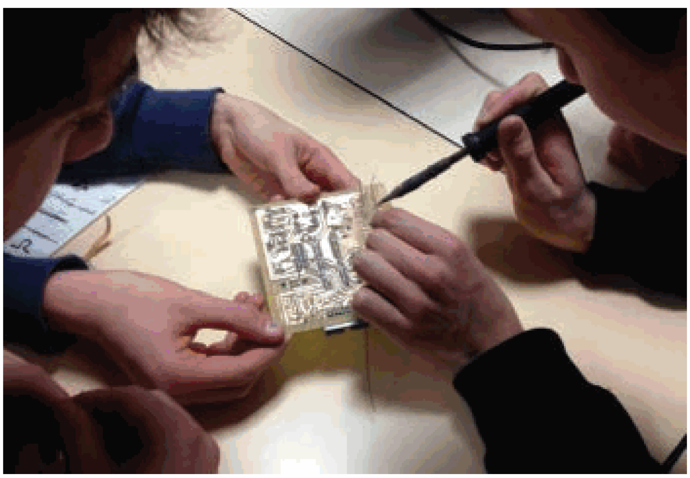


Gypsy in action, on its way to a Gold medal. Gypsy can play the piano and drums, and even curtsy upon completion of its performance.

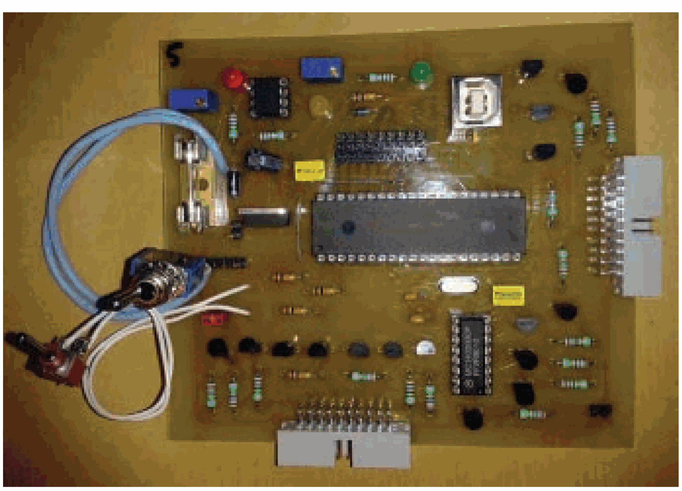
Simulo — four legs, nine motors, six RGB ribbons of LEDs, and three circuit boards.



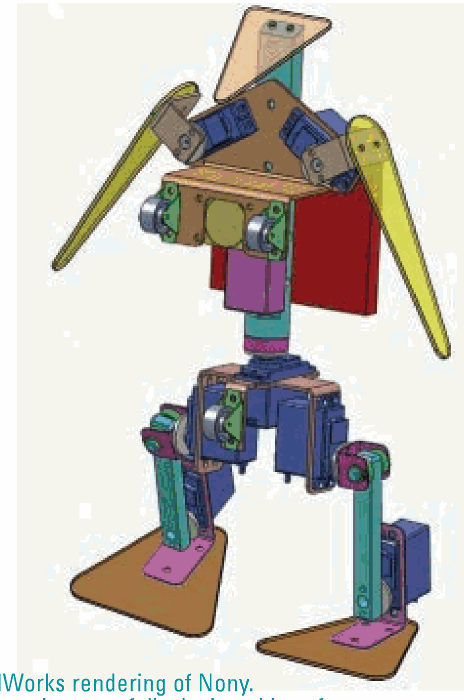
Students learning circuit design with prototyping board.



One of the many custom circuit boards being assembled.



One of the circuit boards for Nony — assembled and ready for use.



A SolidWorks rendering of Nony.
All of the robots are fully designed in software.

The Lycée Colbert team, posing with their robots and medals.

The hard work and attention to detail displayed by the students really showed, and both the students and teachers were very proud of their accomplishments.

For further information on this wonderful program and more pictures of the build process on their robots, go to **http://robotics.colbert.free.fr**. My thanks to Jacques Le Coupanec, Applied Physics Professor at Lycée Colbert, for providing information and photographs for this article. **SV**

Team Kiss My Snake Tears It Up With A Barbie Jeep

by Steven Nelson

Go to **www.servomagazine.com/index.php?/magazine/article/october2013_Nelson** to comment on this article.

Some folks may wonder, what do combat robot builders do when they are not building or preparing for a combat event?

Well, some find other events and technical challenges to keep the creative juices flowing. Many build for and attend the Maker Faire. The Maker Faire has become one of the must-go-to events since it began eight years ago. This year, I became aware of a new event that was to be held at the 2013 Bay Area Maker Faire in San Mateo, CA.

The event is part of the Power Racing series. This event requires teams to build an electric powered race car using small electric kiddie cars like a Barbie Jeep, for example. The teams are allowed to modify the stock machines as long as the original body remains mostly intact and the parts budget doesn't exceed $500. Upon learning about this, I immediately double-dawg dared my friends at the NIMBY space in Oakland, CA to try and beat our team at this race.

Our group consisted of a merger of Team K.I.S.S. robotics and Team Plumb Crazy (Matt and Wendy Maxham) — builders of the combat robot Sewer Snake, among others. Together, we became Team Kiss My Snake (KMS). We have a history of competing with NIMBY and many other teams in another event called Power Tool Drag Racing. So, we knew that NIMBY couldn't resist a new tech challenge.

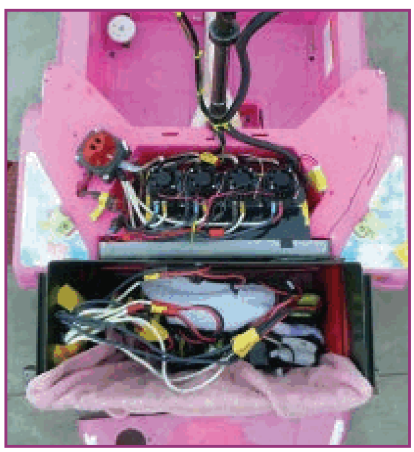
I contacted Matt and we discussed what to use for a drive system. We both have a lot of very used parts from over 16 years of building combat robots. The original design was supposed to use two EV Warrior motors that came from an electric bike and the speed controllers that come with them. We also found an old Barbie Jeep to use.

I didn't hear from Matt for a couple of weeks, so I finally called him and he had already built the car. When I went and saw it, the Jeep had a steel frame with the EV Warrior motors mounted on it, a rear differential/axle, drum brake, the front steering, the actual Jeep body, and, of course, Matt's trademark Red Harbor Freight tires used on his robot Sewer Snake. Matt became busy getting ready for combat at RoboGames 2013, so we didn't do anything for a couple of months.

After RoboGames, I called him about one week before the Maker Faire and Matt said he had some bad news. He drove the racer and both of the speed controllers burned up, so he figured we were done. I did some quick motor/torque math and confirmed the EV Warrior motors

were a bit too small for use with the weight, gear reduction, and top speed we were trying for. I offered two Bosch GPA 750 watt motors and a Vantec RDFR 38E controller. Matt said he had pulled the EV Warrior motors and installed two of my old three inch Magmotor's (A28-400) from my bot, Evelyn a Modified Dawg.

I did a little more math and discovered that with the 10 inch tire diameter and the 7-to-1 gear reduction at 28 VDC, our little Jeep might do 25 mph but it will draw about 175 amps at stall if the machine and the driver's combined weight is about 250 lbs. That's a lot for the Vantec 38E controller in a long race.



Matt said he had several old IFI Thor speed controllers that might work, but they wouldn't play well with the throttle pot/variable resistor he was using. Basically, they needed a servo/PWM signal to trigger them. I told Matt that wouldn't be a problem because I could program an Arduino microcontroller to read the throttle/resistor and send the control signals to the controllers.

So, this started a very long re-build; we had just three days left at this point. The next morning, I wrote the software and tested it with a servo. The software reads the throttle pot as an analog voltage 0-5 VDC, and maps this voltage to a decimal value that is output as PWM values between 1,000-2,000 microseconds. This is fed to the Arduino's *Servo.h* library and then output to the four controllers.

The software also reads a toggle switch that acts as a forward/reverse switch. We tested this late Wednesday with the EV Warrior motors on the workbench and it seemed okay. Of course, there was a ton of work left to do.

The next morning, I made an enclosure for the Arduino Uno we would be using, plus a 9 VDC regulator was added so the Arduino could be powered by 28 VDC off of the main li-ion batteries.

Matt had a lot of wiring from his old robot harnesses, so and he plugged in one of them to connect to the batteries. Nothing happened. The Arduino was DEAD. This is bad. I checked the 28 VDC to the regulator and found the battery polarity was reversed. I think we just fried everything.

We unplugged the Arduino and I checked the 9 VDC regulator with a multimeter. Yep, it was fried. I cut it loose and powered the Arduino with a 9 VDC battery and it still worked. (Whew!) The added regulator acted as a fuse and saved the microcontroller.

At this point, we had Matt go back to finishing the welding and I worked on the logic and control side of the car. The next day, I bought a 9.6 VDC NiMH battery pack and a switch to isolate the logic power from the high voltage. That night, we went back to the build and Matt did the motor and battery wiring while I installed the controls and logic.

For safety, we added a dead man/Matt switch that controlled a 600 amp solenoid from a Team S.L.A.M. robot (Half Gassed) to cut out the main battery power to the speed controllers. This switch must be on at all times for the motors and controllers to be powered. We also added a main battery power cut-off switch. These systems are required to prevent runaway machines.

After another late night, we had a large rat's nest of wiring which connected the Arduino, the controllers, motors, and batteries. Out of seven controllers we had tested, we had four (now working) IFI Thor controllers. Each pair of them powers one of the Magmotors with one controller for each set of motor brushes.

We had one more night to finish. It took until 2:00 AM on Friday night before we actually put the racer on the ground and cautiously turned on the Arduino, then finally the main power. Matt gently tried reverse and it moved backwards. He then tried forward and it moved forward.

We pushed the racer into the street, and Matt hits full throttle, jumps a speed bump and scrapes the steel frame, creating a huge amount of orange sparks as he tears down the street and into the darkness. I was standing next to his wife, Wendy and said, "Well, so much for the first little test." Wendy just smiled.

A couple of minutes later, Matt comes roaring back full speed down the street, grinning like an eight year old with a new toy on Christmas.. Matt shouts, "Man, this thing is fun!" After checking the racer for damage, we noticed it was 3:00 AM and we had to leave for the event at 6:00 AM. So, that was it for testing.

Race Day

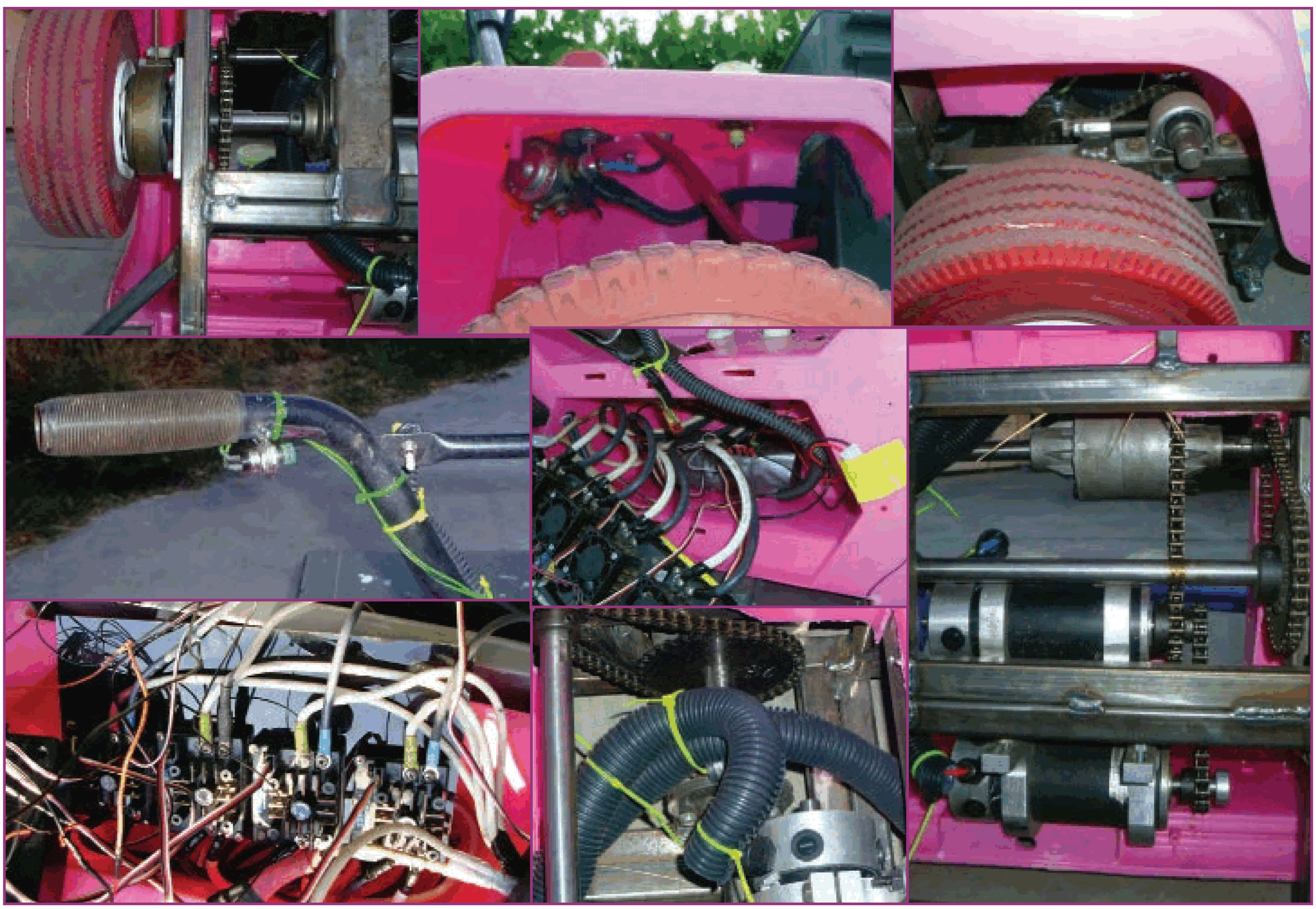
With not much more than 2-3 hours sleep a night for the last three build days, we met up in the pits for the races. Matt told me he had been loaned several more 28 VDC li-ion battery packs from Ray Billings of Team Hardcore Robotics, so we could have about 30 amp-hours of battery power, plus a spare set to keep on the chargers. Wendy provided all of the pit support and the battery management/charging, and Matt and I looked over the racer. Matt had installed dual front tires (since the Harbor freight hand truck tires had bearings designed for maybe 2 mph), plus the race number "666" on the Barbie Army Jeep — so named for the 20 mm ammo can that became the battery enclosure.

In typical no sleep in a week zombie mode, I wandered through the pits and checked out the other machines. Team NIMBY brought a Red Ferrari with a nice steel frame and dual rear tires, a very large starter motor, and Optima SLA batteries. The Make racer (#675) was built using SLA batteries, a large motor, and garden tractor tires. VF (#1) (the only motorcycle) ran very well. Its driver was very tall, so he had to either ride it standing up or with his feet way out in front. I believe it used the original motors. Team Hellematics (#15) used SLA batteries, an Alltech speed controller, and Harbor Freight tires. Team Fun By Q's lightweight racer used SLA batteries, a couple of small 500 series motors (I think Dewalt drill motor size), two vice grips clamped to the steering shaft for handle bars, and some three inch Colson hand truck wheels for tires.

After sizing up the competition, I had some concerns about a couple of them — especially our arch nemesis from NIMBY. I figured the large diesel starter might pump out 2-4 HP at 36 VDC. I knew we had 9 HP and 270 ft lbs of torque available at the rear axle. So, I was slightly concerned. The qualifying laps gave us a chance to see what we were up against, and we felt we would do alright if luck was on our side. Matt added a piece of yellow duct tape to the Barbie Jeep for luck and I added a gray piece. You can never have too much luck in racing.

All the racers were issued an infrared transmitter that triggers the lap timer sensor on the start/finish line. Each trigger emits a unique ID for the timing system. We finally had our chance to qualify and Matt took the first lap kinda slow just to get a feel for the track. Then, of course, he went all out on the second lap and set the fastest time which was 10 seconds quicker than the other cars. That gave us pole position for the first race. We changed the batteries and waited for the first heat.

In the first race, Matt pulled out ahead after the first turn and then began to pull further and further ahead of the pack. NIMBY started having some problems. I think their starter motor got hot. Matt eventually caught up with the rear of the pack and began lapping some of them, then all of them, then he pulled ahead some more, and charged



into the pack like he was behind and lapped them again. He drove so hard that when he got cut off he just ran up against the K-rail barrier and forced his way through the traffic. (This "drive it like ya stole it" attitude is his normal way of robotic combat and racing).

After 15 laps, the Barbie Army Jeep won the first race. We got it back into the pits, celebrated for a few seconds, and then checked the battery, speed controllers, and motor temperatures. The Magmotors were quite hot; the batteries and speed controllers were fine.

We changed the batteries, let the motors cool, and Wendy started charging the used batteries. Then, we noticed some missing set screws from some of the chain sprockets; we were missing three of them. After replacing them, we rested. Just before the second race (while lying on the ground, day dreaming of much needed sleep), I noticed a loose nut on one of the steering rod ends. I tightened it with a wrench. Yep, we better do some nutting and bolting! We turned the racer over and found two more steering bolts missing or loose. This could have been very bad! We replaced them and then checked all of the nuts, bolts, and electrical connections. A good racer/builder knows they must do this after every single run — even if you're dead tired! We got everything tight just in time for the next race. I asked Matt that if he got ahead again in the next race, to take it easy. The poor little Barbie Jeep had never been tested like this.

We mounted a Go Pro camera on his helmet for a first person view of the race and when the green flag dropped, Matt pulled a wheelie and charged into the lead again. (So much for taking it easy.) There were a couple of wrecks in this race, so the yellow caution flag was raised as they were cleared. That only slowed Matt down a little. NIMBY had fried their large starter motor and speed controller in the first race. Amazingly, they got another smaller motor and controller. Unfortunately, at about three laps, it overheated, melting its wiring and burning up their second motor.

After that caution flag, Matt got more aggressive and started power sliding the racer around corners and doing wheelies for the fans in the stands. When this 15 lap race ended, he drove to the winner's circle and did several powered broad slides. He even picked up a rider and drove him around for a bit. I kept thinking, my motors, my poor poor motors!

At this point, the event organizers put NIMBY's driver in a dunk tank and gave Matt three bean bags to attempt a drive-by shot at the dunking target. He hit it on the third try and NIMBY's driver fell into the water.

After this race, we discovered that Matt had destroyed two tires, so we replaced them plus several missing wheel lug bolts. The third race was a combined heat with the electric racers and the human-powered car racers. With some new tires and a new set of batteries, here he went again. The cars are a lot larger and longer than the power wheel racers, so that made passing them very hard on the tight turns of the track. Matt ended up stuck in traffic several times. There were quite a few wrecks and yellow flags, so he didn't win but he still put on a good show.

All in all, we won two 15 lap races and got the highest amount of audience voted "Moxie" points for the day from Matt's performance, aggressive driving style, and of course, for dunking NIMBY. After the race, several of the other teams came to our pits and we explained all of the technology and the math used in building the Barbie Jeep Racer. We also mentioned that the cost of the very used parts probably pushed the $500 limit, but since they were either repaired, recycled, bought as used, or donated by several robotic combat teams and machines, we figured we were in the spirit of the rules. (We would like to thank all of the folks and teams that contributed to this project. It was truly a group effort from the Sacramento, CA valley.)

Matt and Wendy took the Barbie Army Jeep home and we didn't compete on Sunday. Our friends at NIMBY did their own all-niter, rebuilt their motor and speed controller, and completed 75 laps in the endurance race on Sunday, winning the high points championship for the event. (We will get you next time NIMBY ... next time!

What We Learned

The Arduino microcontrollers can easily be adapted to communicate with many — if not all — of the large high current motor speed controllers that are commonly used in radio-controlled robotic combat. The drive train math we were using works for any type of electric vehicle, and allows you to predict the current draws and the battery sizing with confidence. Building machines slightly outside of your comfort zone is definitely challenging, educational, and a lot of fun. Good times ... SV

Experimenting With Machine Intelligence

by John Blankenship
and Samuel Mishal

Go to **www.servomagazine.com/index.php?/magazine/article/october2013_Blankenship** to post comments on this article.

Machine-based intelligence is an exciting field often assumed to be out of the reach of typical hobbyists. This article explores unconventional ways of creating robots that can learn how to solve problems on their own, while adapting to a changing environment.

If you examine dictionaries and encyclopedias, you can find many definitions of artificial or machine-based intelligence. Depending on who you ask, here are a few of the definitions you might find:

- Solving problems through the utilization of sensory capabilities.
- The ability to make decisions based on past experiences.
- Reacting appropriately in the face of insufficient or conflicting information.
- The ability to adapt to a changing environment.
- Behavior demonstrating deduction, inference, and creativity.

The first of these definitions would attribute at least *some* level of intelligence to many of the robots you might see at a robotics club meeting. The last definition implies that intelligence must demonstrate human-like qualities — which allows many hard-liners to argue that machines can *never* attain true intelligence.

The goal of this article is to explore an option in between these two extremes.

We should start by saying that our methodology does not represent the typical AI research, but we think you will find it both entertaining and thought provoking. More importantly, it works.

Our objective is to build a robot that can learn on its own and constantly adapt to changes in its environment. The robot must have a well-defined goal so it can evaluate the effectiveness of its actions, thus providing a basis for modifying its own behavior.

To keep this project manageable, we need a relatively simple goal, yet one that is easily observable. Our robot will learn to follow a line — a common behavior for hobby robots.

Our control program, however, will not contain any line following algorithms. Instead, the robot's internal nature will be to randomly try various actions and evaluate their effectiveness for achieving the desired goal. The ability to perform this evaluation is crucial to the robot's ability to learn on its own.

Evaluating the Robot's Actions

Appropriate evaluation cannot be conducted without associating the action being evaluated with a particular environmental state. For example, it makes no sense to say that *turning left* helps the robot follow a line. We could say though, that *if* a set of line-detecting sensors indicate a specific pattern, then turning left can help achieve the goal. This simple principle will be the basis for our robot's ability to learn.

The robot will constantly observe its environment (by examining sensors) and then randomly perform some movement. If the robot determines that its movement helped achieve the desired goal, then the action taken (and its associated environmental state) will be remembered by storing it in memory.

If the robot encounters the same environmental conditions in the future, then it can retry the action associated with it. If performing the action still helps the robot achieve the goal, then the memory of it should be

strengthened. If the action does not produce desirable results, then the memory should be weakened and eventually discarded. This evaluation process not only allows the robot to determine what works, it also allows it to *change its mind* about what works when the environment changes.

Obviously, the robot must be able to determine if a particular action is helping it achieve its goal. Our robot will have a group of five line sensors for collecting information about its environment. Actions that cause the middle sensor of the group to detect a line will be deemed beneficial because the goal is *actually* being met. If this is the only criteria though, the system would be simplistic at best. What is needed is a way to evaluate if a particular action *helps* achieve the goal, even if that action does not immediately cause the robot to see the line with the center sensor. We will address this problem shortly.

All this sounds complicated, but it is easily implemented in code. Furthermore, the programming itself should help clarify the concepts. A real robot could be programmed to confirm our assertions, but RobotBASIC's integrated simulated robot allows us to experiment in an environment that is easier to control and implement.

Implementing the Principles

Figure 1 shows a simplified version of the RobotBASIC program we used. After some initialization (which creates the environment full of line segments shown in **Figure 2**), an endless loop causes the robot to roam around and observe its environment. The subroutine that performs these actions uses the robot's perimeter sensors which are read with the function *rFeel()* to avoid walls as it moves around. Think of this as a built-in reflex reaction to a hot or sharp object.

When a line is detected using the *rSense()* function, another subroutine is called to react in some way to the presence of the line and analyze the outcome of the actions taken.

Figure 1.

```
main:
  gosub Init
  while 1
    gosub RoamAndObserve
  wend
end

RoamAndObserve:
  // genetically turn away from walls
  if rFeel() then rTurn (140+Random(80))
  if rSense()
    // something got our attention
    gosub ReactAndAnalyze
  else
    if !rFeel() then rForward 1
  endif
return
```

This new subroutine *ReactAndAnalyze* implements the basic algorithm of our program. It is shown in **Figure 3**.

The code in **Figure 3** begins by reading the line sensors and storing them in the variable *CurState*. It then searches the robot's memory to see if that state has been memorized some time in the past. The robot's memory is composed of three arrays: *GoodState[]*, *ActionToTake[]*, and *ConfidenceLevel[]*.

The *GoodState[]* array holds the robot's views of the world (the line sensor data) that it has determined to be important. The corresponding element in the array *ActionToTake[]* identifies what action the robot should take when this line sensor configuration is detected.

The final array *ConfidenceLevel[]* keeps track of how well this memory is working. The use of these arrays will become clearer as we proceed.

Referring back to **Figure 3**, a *for-loop* is next used to search *GoodState[]* to see if it contains the current state of the line sensors. If it does, the variable *InMemory* is set to TRUE, and the value of the corresponding element of ActionToTake[] is used to compute a subroutine name which is then called to cause the desired action to take place.

Most computer languages do not have the ability to *gosub* to a variable name, so if you are not using RobotBASIC, this area of the code could be implemented using *if-statements* or a *switch-case* construct.

After the action is performed, the program must decide if a rewarding result occurred. As discussed earlier, there are two criteria for this evaluation. If the line sensors indicate that the center sensor detects the line, then the variable *Reward* is set to

FIGURE 2.

Figure 3.

```
ReactAndAnalyze:
  CurState = rSense()
   // see if CurState is in memory
  InMemory=false
  if NumStates>0
    for p=0 to NumStates-1
      if GoodState[p]=CurState
        InMemory = True
        break
      endif
    next
  endif
  if InMemory
    // react as expected based on past experience
    // perform the action associated with the current state
    gosub "Action"+ActionToTake[p]
    // decide if the action still produces good outcome
    NewState = rSense()
    Reward=false
    if NewState&2 then Reward=True
    // Also reward if this NewState is in memory
    if !Reward and NumStates>0
      for i=0 to NumStates-1
        if NewState=GoodState[i]
          Reward=True
          break
        endif
      next
    endif
    if !Reward
      // old memory does not seem to be valid
      ConfidenceLevel[p]--
      if ConfidenceLevel[p]=0
        for i=p to NumStates-1  // delete memory
          GoodState[i]=GoodState[i+1]
          ActionToTake[i]=ActionToTake[i+1]
          ConfidenceLevel[i]=ConfidenceLevel[i+1]
        next
        NumStates--
      endif
    else
      // increase confidence of memory (up to 2)
      if ConfidenceLevel[p]<3 then ConfidenceLevel[p]++
    endif
  else
    // try something new
    NewAction =  Random(6)+1
    gosub "Action"+NewAction
    // now see if the New Action produced good results
    NewState = rSense()
    Reward=false
    if NewState&2 then Reward=True  // gave direct reward
    // Also reward if this NewState is in memory
    if !Reward and NumStates>0
      for i=0 to NumStates-1
        if NewState=GoodState[i]
          Reward=True
          break
        endif
      next
    endif
    if Reward
      // good result so add to memory if not there already
      // CurState is what initiated this action so add to memory
      // NewAction is what we did to get rewarding results
      GoodState[NumStates]=CurState
      ActionToTake[NumStates]=NewAction
      ConfidenceLevel[NumStates]=1
      NumStates++
    endif
  endif
return
```

TRUE. The second condition tests to see if the new state of the sensors (the state resulting from the action) is in the robot's memory.

Basically, these two conditions mean the memorized movement will be deemed to produce a rewarding result when *either* a line is detected by the center sensor, *or* any situation is detected that is *associated* with seeing the line.

Every time a memorized action causes a rewarding result, the confidence level associated with this action is increased, but not beyond some preset maximum. If the action does not produce a rewarding result, the confidence level is decreased. If it decreases to zero, then that memory condition is removed from the arrays. This forces the robot to discard situations that do not consistently work, which effectively allows the robot to forget old habits and learn new ones when its environment changes.

Back to **Figure 3**. If the current state of the line sensors is not found in the memory, then the robot performs a random action. When the action is complete, the robot must re-examine the line sensors to determine if the action produced a result that should be rewarded (again using the two criteria previously mentioned).

If either of these criteria is met, then the most recent action is stored in memory along with the environmental conditions that initiated the action. Allowing *both* the actual detection of the line by the center sensor as well as conditions associated with that detection to be considered *good* is a powerful concept that effectively allows the robot to memorize a sequence of actions to accomplish a goal — even though no programming was specifically created in this regard. This principle is very important, so let's summarize the situation.

If the robot performs an action that allows the robot to detect the line with the center sensor, then that action and its associated line sensor state is stored in memory. This allows

the robot to perform the same action in the future if that state is seen again. Furthermore, if the robot ever performs a random action that causes the robot to *experience* states already stored in memory (states *associated* with finding the line), then that action and state is also saved and performed in the future, as well.

The next step is to define the actions that the robot can take. These can be very simple or more complex. The more complex the choices are, the faster the robot can learn.

We can effectively determine the genetics of our robot by deciding what actions we provide for it. The six actions we initially used are shown in **Figure 4**. In each case, the robot moves forward slightly then turns either left or right by a fixed amount. Some actions cause very small turns, while others turn a significant amount. Notice that none of these actions have anything specifically to do with following a line. They could just as easily be used to hug a wall or find a ball. No matter what actions we allow, the robot will use its experiences with its environment to decide which of these actions are important, and store those in its memory.

Using the Program

Due to space limitations, we have omitted parts of the code such as the initialization routines that draw the lines and code associated with using buttons to alter the robot's environment and behavior in real time. You can download the entire source code from **www.RobotBASIC.org** (see the *In The News* tab), as well as a free copy of RobotBASIC. If you run the program, you will see the screen shown back in **Figure 2**. It has many line segments which allow the robot to learn faster because it encounters lines more often.

At startup, the lines are all of medium width, but there are buttons at the top of the screen that allow you to change them to thin lines or thick lines, or even to a mixture of all three types. The top left corner of the screen also displays the number of memories currently in use. As the robot contends with its environment, the number of memories will increase and decrease as things are learned and forgotten.

Each time the program is started, the robot will have no ability to follow a line, but as it roams its environment it quickly learns what works and what doesn't. Often the robot becomes reasonably adept at line following in 30 seconds or so, and it gets better (especially at acquiring the line) as time progresses.

If you run the program multiple times and look carefully at each robot's behavior, you will see that they seem to have different personalities. Some of the robots will follow the line by staying centered as you might expect. Some though, will stay on the left side of the line while others will stay on the right. It all depends on what the robot encountered during its learning cycle and what actions it randomly took.

Once the robot has learned to follow a particular line width, press one of the buttons at the top of the screen to change the width of the lines. Sometimes the robot will be able to handle the new lines right away, but often, the actions stored in the robot's memory will not work well with a different line width — especially on sharp turns. In such cases, the robot will quickly forget the actions that do not work and learn new ones.

Modifying the Code

Experimenting with various alternatives can be intriguing. In addition to changing the width of the lines, you could change how quickly the robot forgets (just change the maximum allowable confidence level) in order to see how that affects its ability to adapt. With a little modification, the program could even allow the robot to change its own propensity to forget based on a long-term self-evaluation of its efforts.

Figure 4.

```
Action1: // easy left
  rForward 1
  rTurn -1
return

Action2: // easy right
  rForward 1
  rTurn 1
return

Action3: // medium left
  rForward 1
  rTurn -3
return

Action4: // medium right
  rForward 1
  rTurn 3
return

Action5: // Hard left
  rForward 1
  rTurn -6
return

Action6: // Hard right
  rForward 1
  rTurn 6
return
```

You could also change the nature or number of the built-in behaviors. One of the most interesting things we tried was substituting the behaviors in **Figure 5** for the last two behaviors shown in **Figure 4**. The actions in **Figure 5** give the robot an advantage over the original version.

The robot moves forward a bit, but then turns to the left or right a maximum amount of up to 70º, but

Figure 5.

```
Action5: // hard left
  for i=1 to 10
    rForward 1
    if rFeel()then break
  next
  for i=1 to 70
    rTurn -1
    if rSense() then break
  next
return

Action6: // hard right
  for i=1 to 10
    rForward 1
    if rFeel()then break
  next
  for i=1 to 70
    rTurn 1
    if rSense() then break
  next
return
```

stops the turn if it sees the line with *any* of the sensors. To some extent, these new actions give the robot a very limited form of vision because the robot can *look* for the line rather than just performing an action and hoping it will find the line.

This ability to look around for the line is much more efficient than just turning a fixed number of degrees, and robots capable of these actions learned much more quickly. They also learned ways of following the line far different from anything we might have imagined. They often follow the line by just staying close to it (instead of being centered on it). Oddly too, these robots often shift from one side of the line to the other as they move along it — an unexpected and very unusual behavior. To make it easy for you to observe these options, the program has buttons that allow you to make the robot forget easier and remember longer, as well as buttons that allow and disallow the vision-based actions shown in **Figure 5**.

Most of these buttons have no effect on the robot's memory. When vision is added or removed, however, the memory is cleared so that the new criteria can be learned. There is also a button that displays the contents of the robot's memory so you can scrutinize what has been learned.

Practical Considerations

Real robots could certainly employ these principles, but generally they would move much slower than the simulation and take far longer to initially learn. You could easily save the array data from the simulator though, and use it with a real robot allowing it to take advantage of everything learned by the simulator. The real robot could still adapt by continuing to learn and forget — just as the simulator does — but without having to go through the initial learning phase.

The program could easily be adapted to allow the robot to learn different things such as how to hug a wall or find a ball by giving the robot new criteria for determining when the goal has being satisfied. Of course, changing the nature of the goal might mean you need to add or change the sensors your robot has. A more advanced program might have multiple goals, allowing the robot to deal with a much more complex environment. We examined such a prospect in our book, *Robot Programmer's Bonanza*.

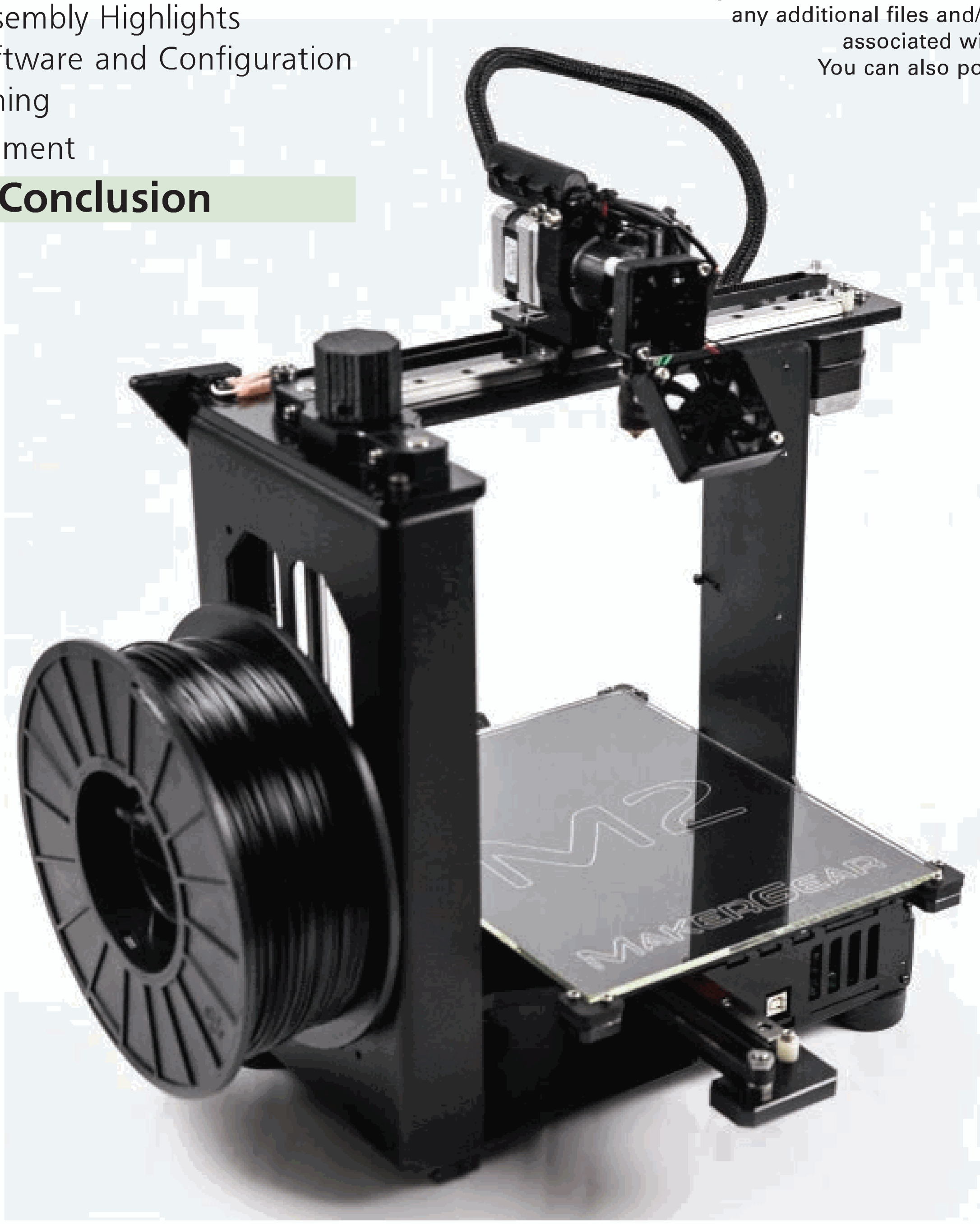
Even though these techniques are unconventional, we have shown them to work exceedingly well and we encourage others to experiment with them to see how they might be used with real robots. **SV**

3D Printers

This month:

by Michael Simpson

Go to **www.servomagazine.com/index.php?/magazine/article/october2013_Simpson** for any additional files and/or downloads associated with this article. You can also post comments.



Using the five printers covered in this series over the last few months has really given me some insights into what to look for when purchasing a 3D printer. I will share this insight, but first I want to summarize my experiences with each printer.

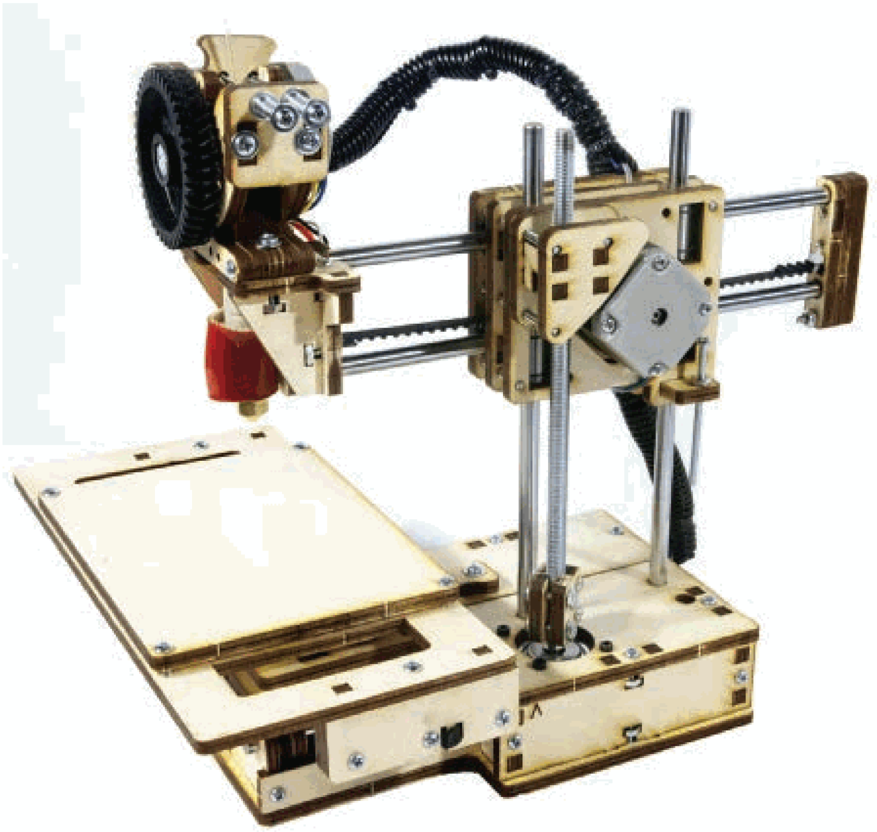
3D Printer Summaries

When I first started researching 3D printers, I wanted a way to compare the build size to printer cost. So, I created the formula **(X * Y) / cost * 1000** which I'm calling the Build Cost Ratio or BCR. The higher the number, the better.

Keep in mind that the number does not consider other aspects of the printer such as features, rigidity, or print quality. For example, the Rostock Max has a BCR of 78, yet it is the worst printer in the bunch in my opinion.

Where the number really helps is when comparing printers with a print quality in close proximity to one another. For example, let's compare the Afinia to the Maker M2. Both printers have excellent print quality, so the BCR is very relevant. The Afinia has a BCR of 15 and the M2 kit has a BCR of 54.

FIGURE 1.



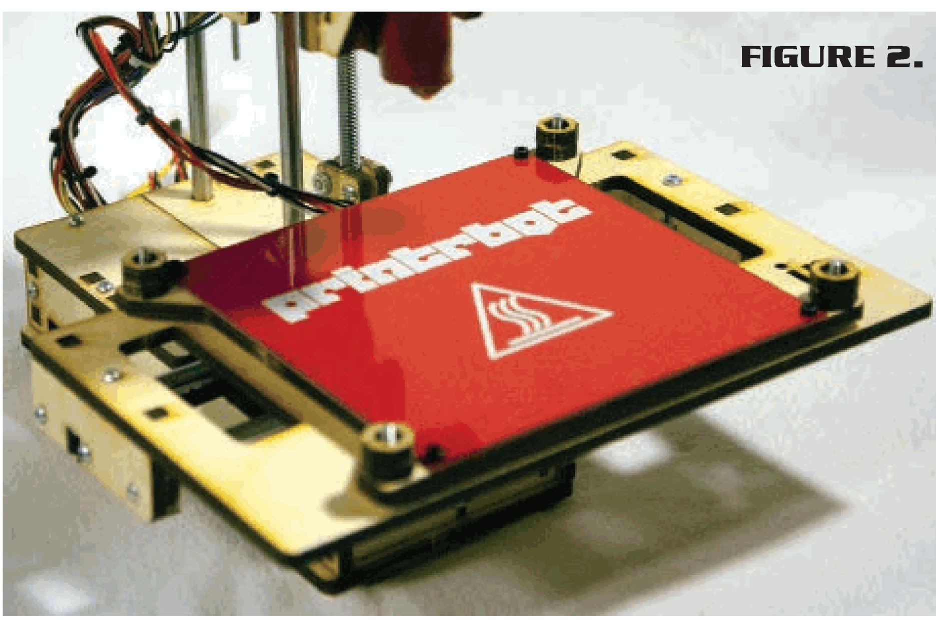


FIGURE 2.

PrintrBot Jr Summary

I have to admit the first thing that attracted me to the PrintrBot Jr (**Figure 1**) was the price. At $400 in kit form, it is the lowest priced printer in its class. It has a BCR of 60.

The printer was easy enough to build, but being made from wood it lacked the rigidity required for consistent quality prints. Sure, you will be able to create a print, but you have to constantly fiddle with the bed adjustments to keep it printing.

The controller used on the PrintrBot Jr is also not my favorite. Even using the settings outlined on the manufacturer's website, I have yet to get the printer calibrated to a point where a print is dimensionally correct.

An upgrade that you will need to purchase is the heated bed shown in **Figure 2**. The heated bed will allow you to print ABS and have better control of printing PLA.

While these low priced 3D printers are nothing more than novelty items, I leave it up to you if the $400 price tag is work dabbing your toe into the 3D printer waters. Even after spending $100 for the heated bed upgrade, you are likely to get your toe bitten off if you are not careful.

Please be sure to post any questions in the *SERVO Magazine* forums at **http://forum.servomagazine.com/viewtopic.php?f=49&t=16968**. I will also be posting additional information on my website at **www.kronosrobotics.com/3d**.

My PrintrBot Jr stopped working, and while I have yet to troubleshoot the printer, I suspect the problem is somewhere in the nest of wires shown in **Figure 3**.

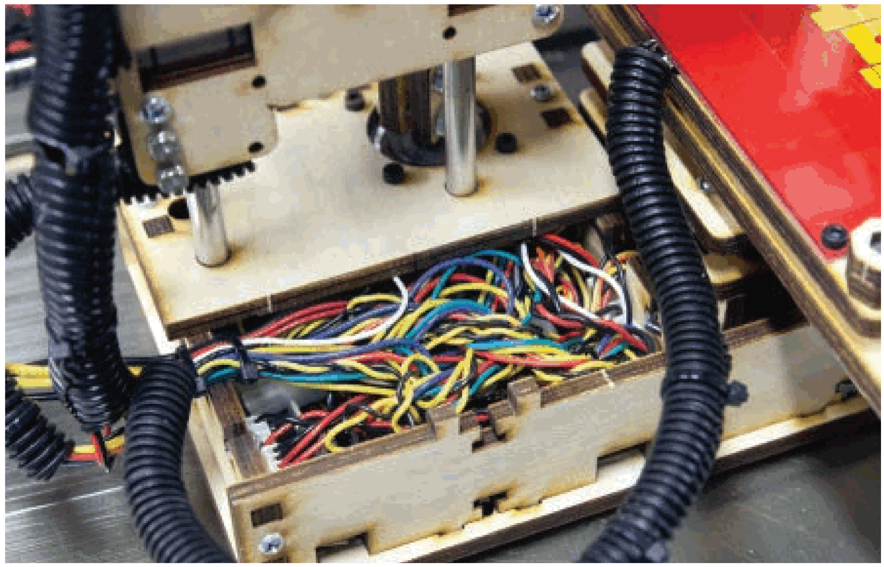


FIGURE 3.

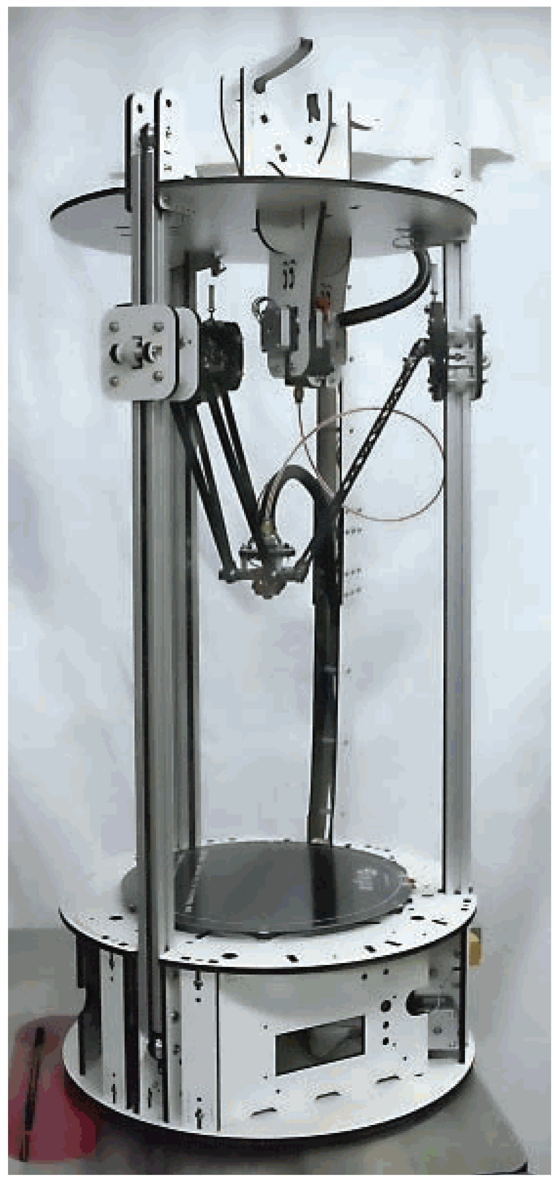
Rostock Max Summary

At first glimpse, the Rostock Max looks like a great printer. It looks cool and has a BCR of 78. To be honest, I have had nothing but problems with the Rostock (**Figure 4**). If it were a $400 printer, I might cut it a little slack, but with a $1,000 price tag, its prints are not even on par with the PrintrBot Jr.

Now granted, most of the problems can be attributed to the overly complicated extruder system, but there are other problems that plague this printer. For instance, the heated bed is underpowered and has a hard time reaching 70°C, much less the 100°C I like to use when printing with ABS.

There are other problems, as well. Its poorly thought-out homing switches consist of a screw (**Figure 5**) that must be inserted at an angle in order to make contact with the switch. Delta printers are hard enough to get the bed leveled, but this makes it almost impossible to get consistent results.

FIGURE 4.



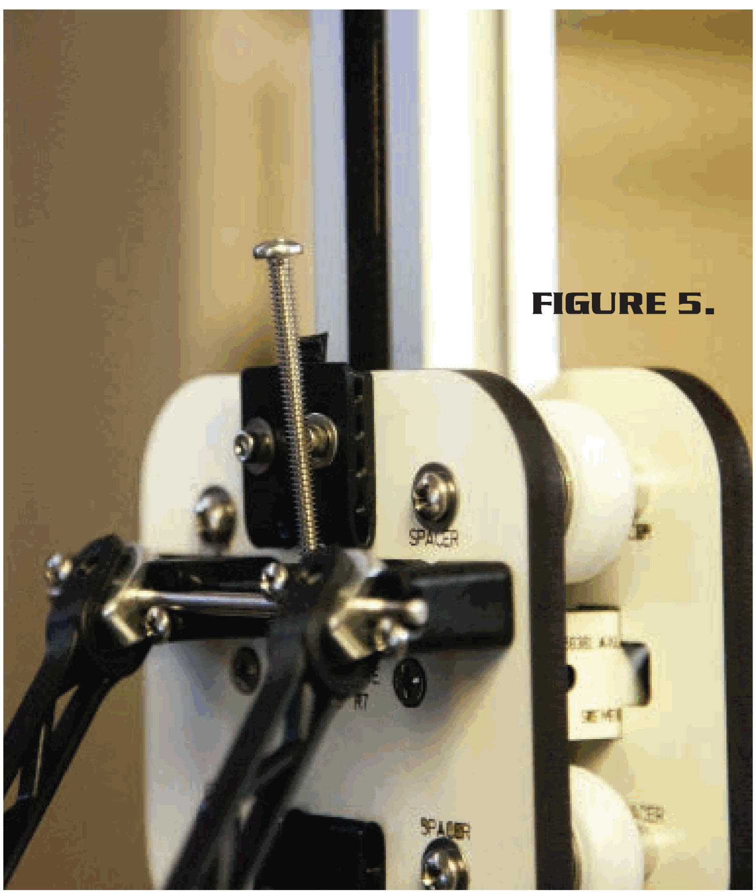


FIGURE 5.

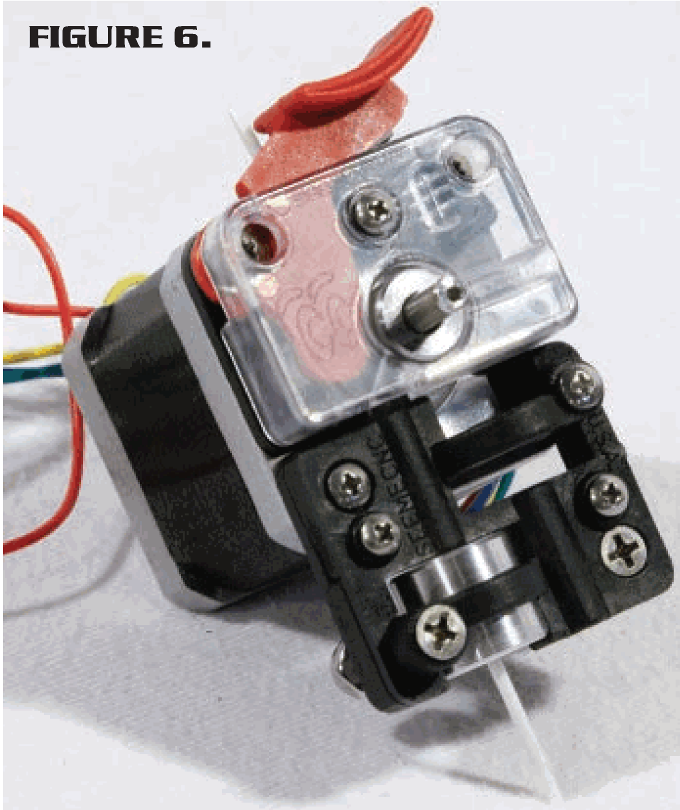


FIGURE 6.

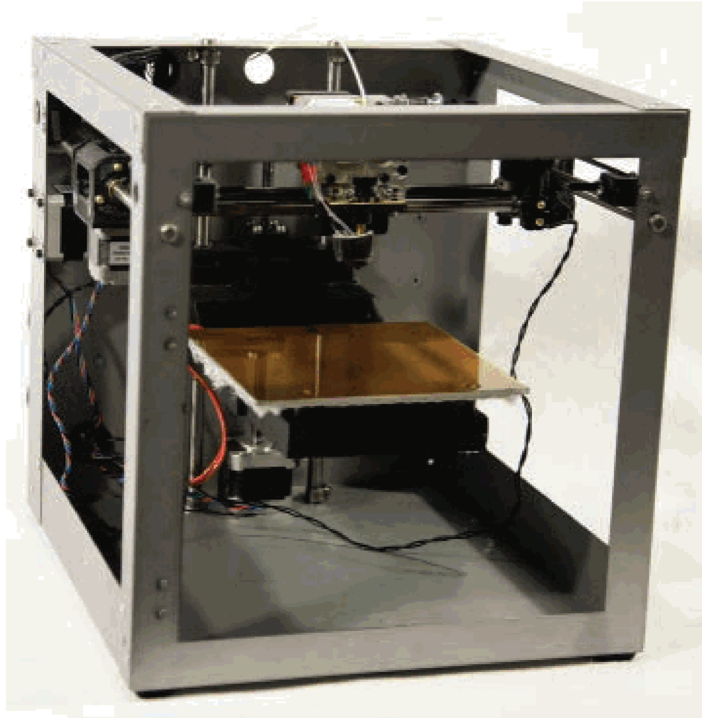


FIGURE 7.

To give a little credit back to the Rostock folks at SeeMeCNC, they have come up with a new direct drive extruder shown in **Figure 6**. This is a much simpler extruder that is easy to build and install. I have two of them installed on my machine, but getting the new settings for this extruder is like pulling teeth. This leads me to the major issue with the Rostock Max. There just seems to be no cohesion as a company. For example, they seem to rely on others in the forums to support their products.

If you are looking for an experimental 3D printer that is still in beta, then the Rostock Max may be the printer of your dreams. If not, then continue reading as my 3D printing experiences do get better.

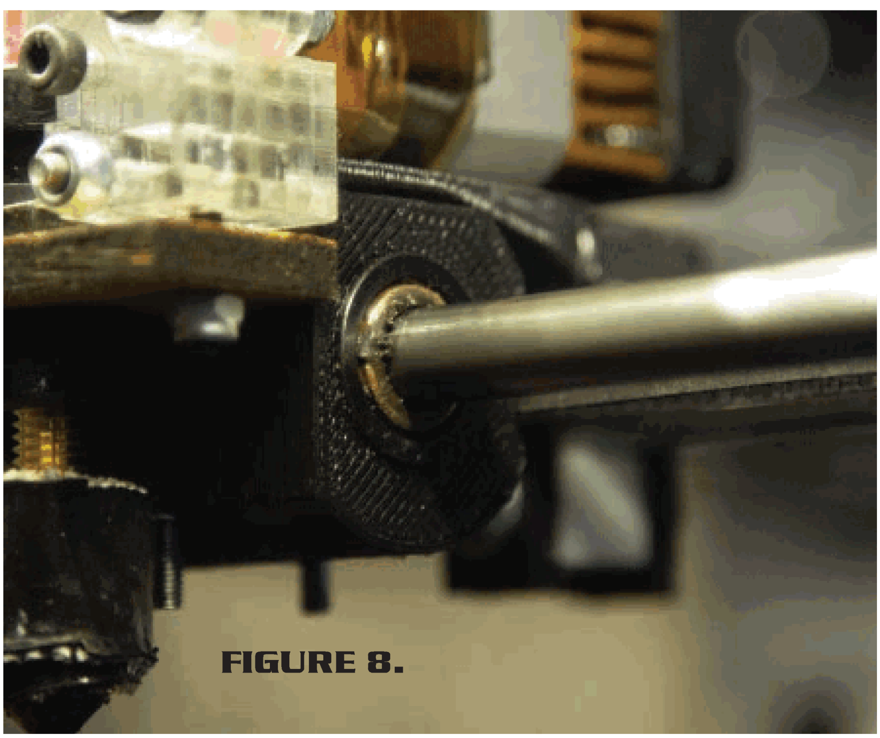


FIGURE 8.

Solidoodle 2 Pro Summary

At $599, the Solidoodle 2 Pro (**Figure 7**) is starting to get you closer to a 3D printer that can produce prints consistently enough that you can use it for prototyping your designs. It has a BCR of 60, and features a steel frame that is more rigid than either the Rostock Max and PrintrBot Jr. It has a better extruder and with some tuning, it can create consistent prints. It isn't without its problems, however.

The Solidoodle 2 Pro does not use bearings for its linear motion, but instead uses brass/plastic (**Figure 8**) bushings. Even after tuning them, I found they would require further attention after even a little use.

While the Solidoodle 2 Pro has an aluminum build platform, it is supported by a wood base (**Figure 9**). They did a good job at concealing this fact by coating the wood in block plastic, but the lack of rigidity shows through.

The heated bed is a little under-powered, and the highest temp I could achieve was 95°C. This is just shy of what is needed for good adhesion on Kapton.

I have created prototype parts on my Solidoodle and with a little work, it could be turned into a very consistent printer.

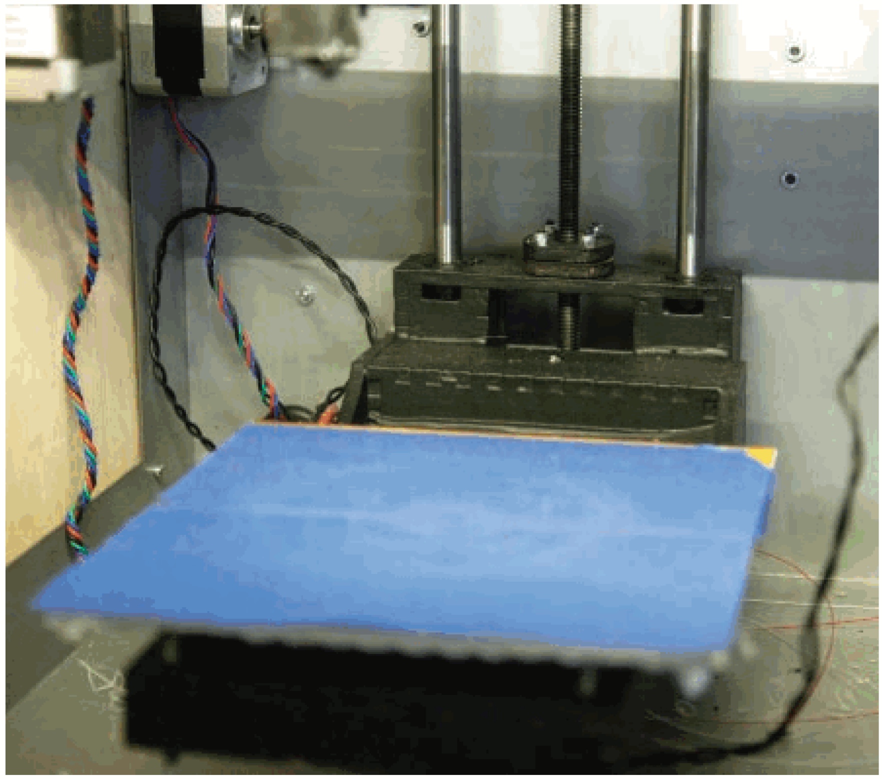


FIGURE 9.

Afinia Summary

The Afinia printer (**Figure 10**) has received my praise since the day I received it. Simply put, it provides great prints right out of the box. Some of the early problems I had with the printer going crazy after a print completed have been fixed by a software update.

The Afinia's unique use of a perforated board (**Figure 11**) as a build platform works very well. That — coupled with the best rafting and support generating system — makes for consistent trouble-free prints time after time.

However, all is not golden. The Afinia does have some issues. First, you can't totally turn off supports. This can be problematic at times. The lack of low level print options can also be an issue. The closed source software has opted for a much more simpler approach to printing objects.

While the Afinia can print PLA, it is not a PLA-friendly printer. The extruder design does not do a good job at keeping the filament cool in the top portion of the extruder. In addition, the Afinia does not have a spring loaded tensioner. On more than one occasion, this has caused PLA to jam up the extruder. This issue is so bad that I have given up printing PLA on this printer.

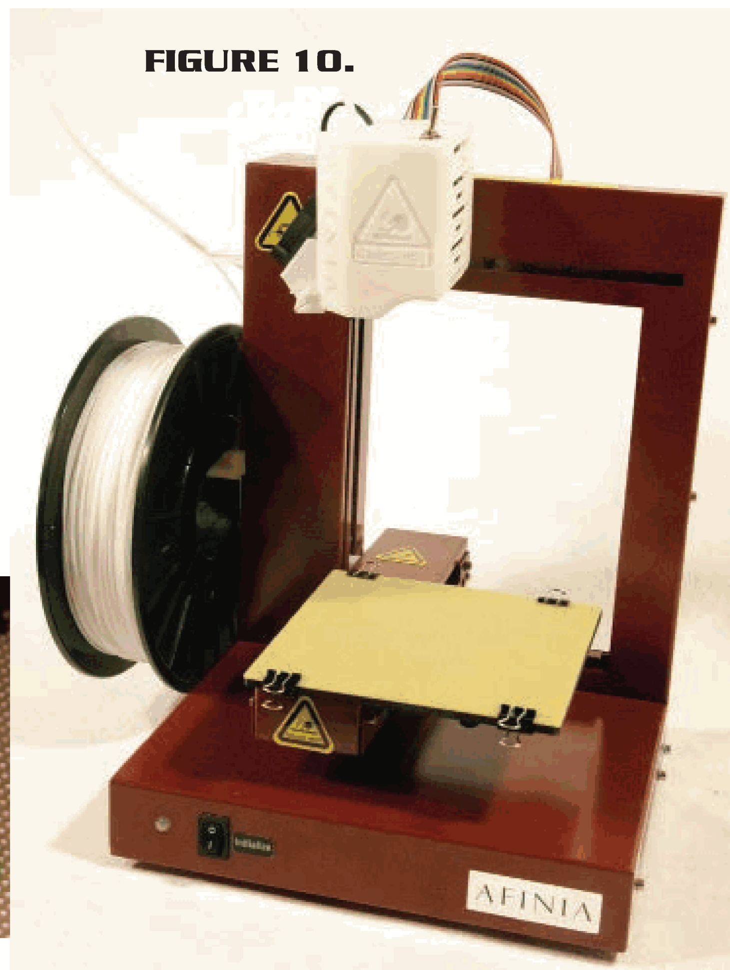


FIGURE 10.

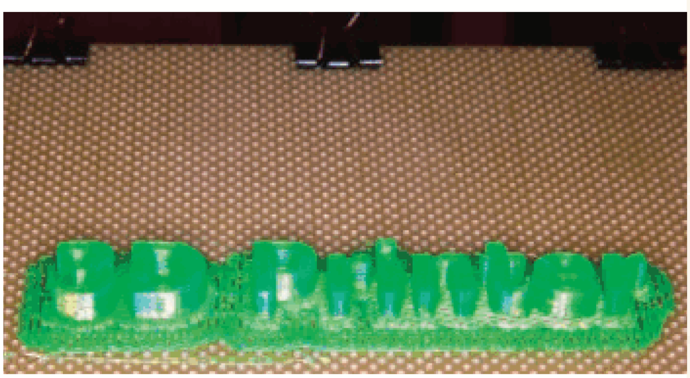


FIGURE 11.

The operation of the Afinia is tied closely to its own proprietary software. Unlike all the other printers in this series, the Afinia cannot use third party software. While not necessarily a problem now, software like Creator is rapidly catching up with the foolproof printing abilities of the Afinia.

The last issue with the Afinia is the print volume to price point. The Afinia has a 5" x 5" x 5" build area, and with its high cost yields a BCR of 15. This may turn many potential buyers away. That said, if you are only going to print small parts, no other printer comes close to the Afinia's capability in producing small prototype parts.

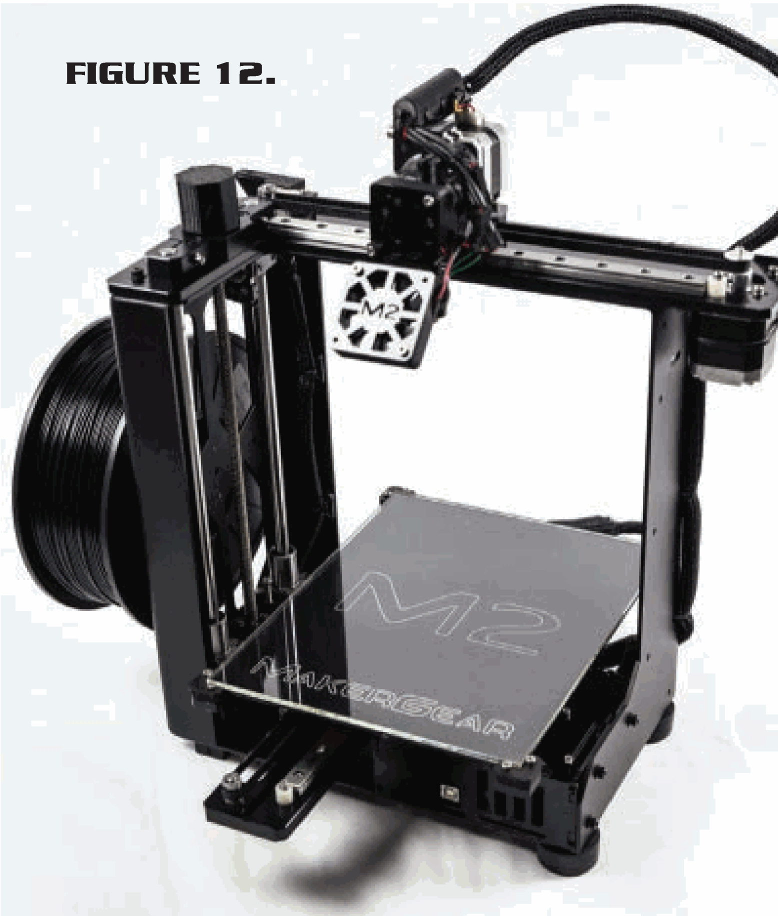


FIGURE 12.

MakerGear M2

The MakerGear M2 3D printer kit arrived about half way through this series. Since then, this has been my go-to 3D printer even though the M2 was the most expensive printer in this series. Priced at $1,775 assembled and $1,475 in kit form, its generous 8" x 10" x 8" build platform yields a BCR of 45 and 54, respectively.

The M2 (**Figure 12**) features a removable heated bed. It uses a standard 8" x 10" profile, so framing glass or acrylic can be placed on the platform support. I have printed PLA on cool acrylic, heated glass, and Kapton tape, and ABS on heated Kapton at 110°C — all without issues. The bed is easier to level than any other printer I have used to date. The M2 is by far my favorite printer, but I did have to tweak a few things.

Soon after putting the printer to use, the main extruder cooling fan burnt out. The M2 comes with two power supplies (**Figure 13**). The first is a large 12V power supply that is meant to power the heated bed. The second is a smaller 19V power supply that is meant to power the logic, motors, fans, and extruders. The problem with this configuration is that all the fans and extruder use this power source. Most 12V fans will not last long on 19V.

By changing the wiring a little on the power connectors, I was able to share the 12V for the heated bed with the logic and MOSFETs. I kept the motors on the 19V supply. This worked perfect. The new fan runs cool and the 5V logic regulators don't run as hot. The extruder takes just a little longer to heat up, but its curve is more predictable. Problem solved.

I also had an issue when printing PLA which would ooze and bubble, and the prints were not very good. I purchased one of those $20 meters with thermocouples so I could measure the actual temperature of the hot end. I found that the hot end was running several degrees hotter than what was being reported.

There are a couple things that can cause this problem: bad settings in your firmware or a thermistor that isn't making contact with the hot end. It ended up being the thermistor. I added a dab of heatsink compound to the thermistor and more Kapton tape. Problem solved.


FIGURE 13.

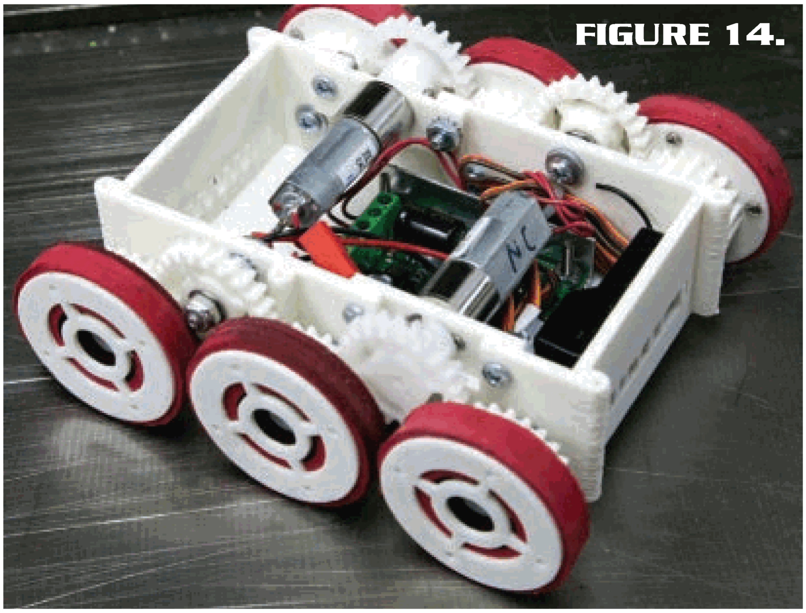


FIGURE 14.

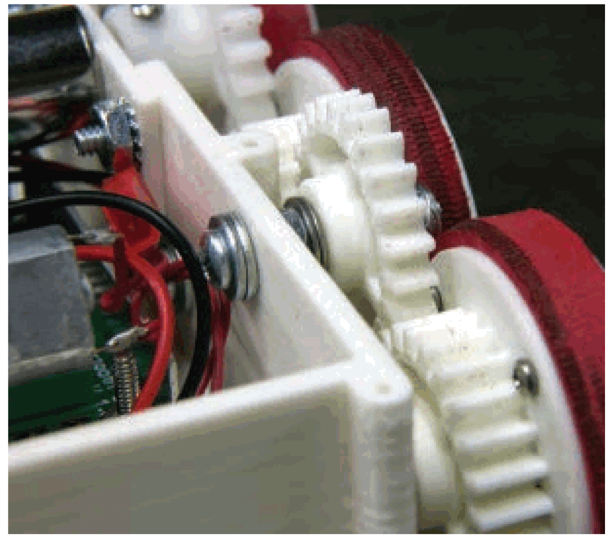


FIGURE 15.

Finally, I added an LED strip to the top of the printer to help light things up when the printer is running.

The MakerGear M2 uses a Rambo controller, so by printing a new extruder mount it would be very easy to add a second extruder to the printer — so easy, in fact, that with the open design you would only lose about an inch on the X axis. This will definitely be one of my future projects. Both Slic3r and Creator support multiple extruders.

The MakerGear M2 is the printer to beat, but the proof is in the pudding. **Figure 14** and **Figure 15** show the base for a fully 3D-printed battle bot that we will be building in an upcoming series. All the parts were printed on the M2.

MakerBot Replicator 2 and 2X

I have had the chance off and on to play with both the Replicator 2 and 2X. While the early versions of the extruder were a nightmare and the software is just now getting to the point where it's on par with the Afinia or Creator, they are both good printers and up there with the MakerGear M2 and the Afinia.

The Replicator 2 sports an 11.2" x 6" build area, but at a cost of $2,200 it has a BCR of 30. It does not have a heated bed and you cannot upgrade the printer to use one, so you can't print with ABS. For this reason, I can't recommend this printer — especially at the price they are asking.The Replicator 2X sports a 9.8" x 6.3" build area and a price tag of $2,800, but only yields a BCR of 22. However, keep in mind that the Replicator 2X does have a couple of extra features.

First, it sports a second extruder. This is why the build area is smaller than the Replicator 2. It also has a heated bed so you can print with other materials, which can be real handy with two extruders. Even with those features, is the Replicator 2X worth $2,800? It depends on your point of view. While I can add a second extruder very easily to the MakerGear M2, the 2X is ready out of the box to print with two extruders. For those with a lot of disposable income, I'm sure this is just what they are looking for.

How to Select Your Next 3D Printer

Okay, you want to purchase a 3D printer, but new ones are presented on Kickstarter every month. How do you decide? I have purchased five printers in the last six months and unfortunately have been burned. Here's some advice that comes from my personal experiences:

1. Take what the manufacturer says with a grain of salt.

You need to treat the manufacturer of a 3D printer like a used car salesman. They want to sell you a 3D printer; they will say what it takes to sell you a printer. While they may give you specs on the printer, getting it to print at those specs may be difficult or impossible. You are better off looking at other sources for some of the printer information. You can search online for forums and reviews of the printer you are considering. Be sure to search YouTube. Other sources of information are local clubs or 3D printer stores.

2. Beware of what you may see on YouTube.

When you are researching a particular printer, be sure to do a search on YouTube. If you see no examples of the printer you are looking for, there is a chance that it is a very new printer design and no one has uploaded a video. That said, any manufacturer that sells a 3D printer should have videos on YouTube and links to videos on their website (See Tip 5). If they don't, this should raise a red flag. If you

do see videos, are they all from the manufacturer or from actual printer purchasers? If you see a lot of videos from several different users and they are mostly positive, then that's a big plus. If you see more than a couple of bad reviews, then that's a very bad sign. If you see a mixture of both good and bad reviews, it may indicate that the manufacturer has poor quality control.

Here is the big one to watch out for: It is possible to print specially designed objects that don't require retraction. Things like tank treads and stretch bracelets will print almost perfect on even the worst printer. This is because the printer does not have to do anything more than extrude a single thin line of plastic over and over as it prints each layer. Remember, proper retraction is what really separates a crappy 3D printer from a good one. Almost all objects require the use of retraction to get a quality print.

3. Look for metal frame printers only.

Look for a printer that has a steel or aluminum frame. This can be covered in wood in plastic, but the portion that supports the linear components should be metal. Stay away from wood, plastic, or melamine based printers. They lack the rigidity to give you consistent and accurate prints.

4. Look at the linear motion components.

The linear motion components should all use bearings. If using steel rods and linear bearings, the rods should be larger than 8 mm for printers with spans greater than six inches. Actual linear rails are better than rods and bearings. Linear rails will add additional cost to the 3D printer but will make the printer more rigid and accurate. Watch out for printers that use plastic or wood on the X, Y, or Z carriages.

Look for G2 belts for driving the X and Y axis, and look for tensioners on those belts. Stay away from ball screws or ACME screws on the X and Y axis. They are too slow and problematic when used with 3D printers. Finally, look for an ACME or ball screw on the Z axis, and watch out for the standard threaded rod used on some cheap printers.

5. Study the manufacturer's website.

Go through every inch of the manufacturer's website, in fact. The site should have technical information about the printer you are researching, and there should be lots of pictures of the printer in action with links to videos. Look for a setup and configuration page. A link to an FAQ page or forums is a real plus. The support page should not just point you to forums. Contact information should be available for support via phone or email.

Look for replacement parts listed on the website. If they don't sell them, how will you fix your printer? Most — if not all — manufacturers don't offer warranties on their products.

6. Take a close look at the extruder.

Look for a direct drive extruder. You want a printer that's as simple as possible. The direct drive extruder should have some sort of adjustable spring-loaded bearing that puts pressure on the filament. Look for an extruder that has easy access to the drive gear so you can clean it from time to time. Stay away from printers with Bowden cables. If you can't find any information on the extruder, then contact the manufacturer via email. If they don't answer, look for a different printer.

The printer should have two fans mounted on the extruder: one is for cooling the filament just above the hot end and the other is for cooling the extruded filament.

7. Only purchase a printer that has a heated bed.

You want a printer that comes with a heated bed or — at the very least — a heated bed option. The bed should be capable of reaching temperatures of at least 100°C. Some manufacturers sell their printers as PLA printers so they don't have to offer heated beds, resulting in a simpler or cheaper design. The thing is, a heated bed gives you more options for how you print PLA. PLA will stick to glass when it's heated to about 70°C, and it will pop off when the glass cools.

Heated beds require bigger power supplies. The larger the print bed, the larger the power supply.

8. Take a close look at the controller.

A good controller will allow you to upgrade or reconfigure your printer. You will be able to add things like extra cooling fans and heated beds. Some controllers — like the Rambo — allow you to add a second extruder. Some closed-source printers have proprietary controllers which means they are what they are, and you can't make changes or add upgrades later.

To add a second extruder, you need the following:

- Five stepper drivers
- Three thermistor circuits
- Three MOSFET controls minimum (ramps), or five (Rambo) if you want to control the speed of the cooling fans

Currently, the Rambo controller board is the cream-of-the-crop 3D printer controller board.

9. Look into the upgrade options.

Check out what options the manufacturer offers. Normally, upgrades offered by the manufacturer will be more refined than upgrades you do yourself.

Ones to look for are:

- Fan upgrades
- Heated bed upgrades
- Dual extruder upgrades
- Power supply upgrades
- Covers or enclosure upgrades
- Software upgrades

Please note that some printers come with all or most of these upgrades, so they won't be offered on the website.

10. Don't buy a multi-use machine.

Don't buy a 3D printer that can also be used as a CNC, or a CNC that can be used as a 3D printer. They are two different machines and have different needs to produce good results. A 3D printer will require fast movements with almost no load. A CNC requires mass and can place a significant load on the machine.

11. Look at the lead time for printer delivery.

Look at the time it takes to get the printer. A longer lead time indicates a smaller company that is running in bootstrap mode. That means they take your money, *then* order the parts needed to put the machine together. These companies are constantly playing catch-up. That said, my M2 had a six week lead time and it is one heck of a printer. I have seen some companies with lead times of over two months. Shy away from these.

12. Check out Thingiverse.

Do a search on Thingiverse for the printer you are researching. If the printer has been available for a while, this site may reveal problem areas and upgrades that fix them (or not).

13. Stay away from Kickstarter.

I'm a big fan of Kickstarter, but with things like 3D printers your money may be tied up for over six months. There also is no guarantee you will ever get the product, or the printer will still have kinks to be worked out.

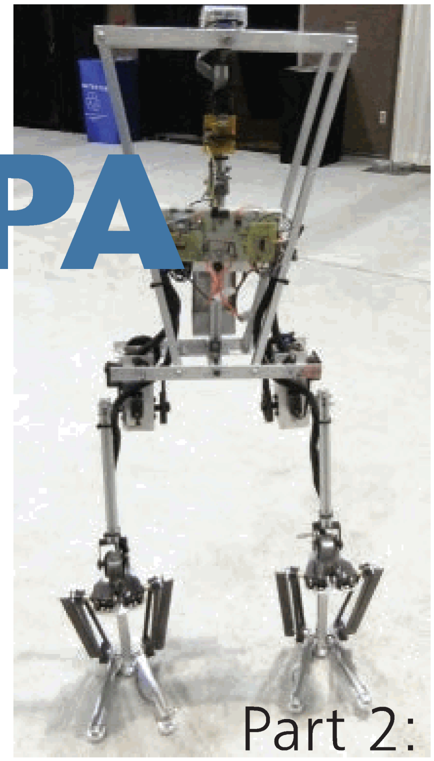
Final Thoughts

The bottom line is this: If it sounds too good to be true, it probably is. Do your research because it's a jungle out there when it comes to 3D printers.

As for my recommendation, if you want to purchase a quality 3D printer to create prototype parts, look at the MakerGear M2. If you know you won't ever print anything larger than four inches or so, then look at the Afinia. **SV**

I will be starting the 3D-printed battle bot series very soon — hopefully with the November issue. We will use our 3D printer to create a fighting bot. The only rule is it has to be printed in plastic. You can see the platform in a video at **www.youtube.com/watch?v=pK57Tw3dL3U**.

The Road to the DARPA Robotics Challenge



Part 2: Mechanical Design

by Daniel Albert and Chris Mayer

Go to **www.servomagazine.com/index.php?/magazine/article/october2013_Albert** to comment on this article.

If there are any die-hard *SERVO* readers out there, you may recall the four part article back in 2011 about the "Big Walker." Well, a great deal has changed on this biped since then. Only about 10% of the original parts remain. Most of those are the servos and they are about to be replaced, too. There are a few pieces of angle aluminum, controller board, and the load cells. Pretty much, that's it.

Similar to the efforts of the famous self-educated clockmaker John Harrison, the big biped project is an attempt to build a precision mechanism that differs from the standard accepted designs. Novelty and imagination are the key ingredients. Solving problems is a reiterative process of analysis and rework. This type of project takes years of hard work. Disappointment is common.

As we get closer to our goal of a stable system that meets our constraints, we are thankful that we did not merely go down the path of the standard accepted design of square flat feet. We will leapfrog the other heavy footed designs and obtain a working biped that mimics nature in efficiency and elegance.

This is a very complex project requiring several engineering disciplines. While we are both professional software and firmware guys, it has been a learning experience in mechanical design and fabrication. We are learning mostly by trial and error that what works on paper doesn't always work in practice. This has prompted us to rebuild the biped a third time.

We discovered that the second design constructed two years ago would not work. Here is why.

In order to walk well, a biped must be able to support itself on one leg. The first two designs could not. It was initially presumed that if most of the weight could be translated to one side, that the opposite leg could be moved into a more forward position to catch the falling mass. This is a dynamic gait similar to the way humans walk. Like a pendulum, the mass would swing to one side and then back.

As humans, we throw our weight outside our center of mass and fall towards a determined spot where we place our foot to either stop our movement or transfer our mass to a new vector. This works in theory but when put into action, things get more difficult. We lost control of the system due to low power, slow reaction time, and lack of stiffness. We had trouble just with the one leg stance. Each attempt of moving from a two-legged stance to one side yielded different results. Our rigid body dynamics analysis proved incorrect. The truth is "nothing is rigid."

Power is crucial for holding the given mass of the biped from collapsing. Servos are always rated higher than their actual stall torque. The servo's rated at 400 oz-in of torque turned out to provide about one half that in a pure stall. The movement from two legs to one caused the power consumption to spike. The servos got hot and soon went into current-limit protection mode; they shut down and the biped collapsed.

The pulley reduction designed in the first system does protect the servos from damage and creates more torque, but even a 2:1 increase in torque was not enough.

It was presumed that a fast reaction time to the moving mass could keep the biped in positions where the stress on individual servos was minimized. This helps, but cannot replace needed power. It takes many mSecs of time to change the direction of a servo. It has rotational mass that may need to reverse.

This isn't helped by the very slow 50 Hz control time of

the hobby R/C PWM control pulse. Put 20 servos together in one system and without a neural net that can learn to adapt, all you can do is a slow static frame system.

Stiffness is one of the most important pieces of the puzzle. If a joint is designed to rotate on one axis and because of slop it moves slightly in another, then the system becomes less deterministic. Slop is almost always uncontrollable. It is unactuated and very hard to predict. There may be a time in the future when we use unactuated movement to enhance a complex system (like the flight of a bat or an F-117) but for now, "if in doubt ... leave it out."

This last rebuild removed approximately 80% of the slop in the original design.

The Changes

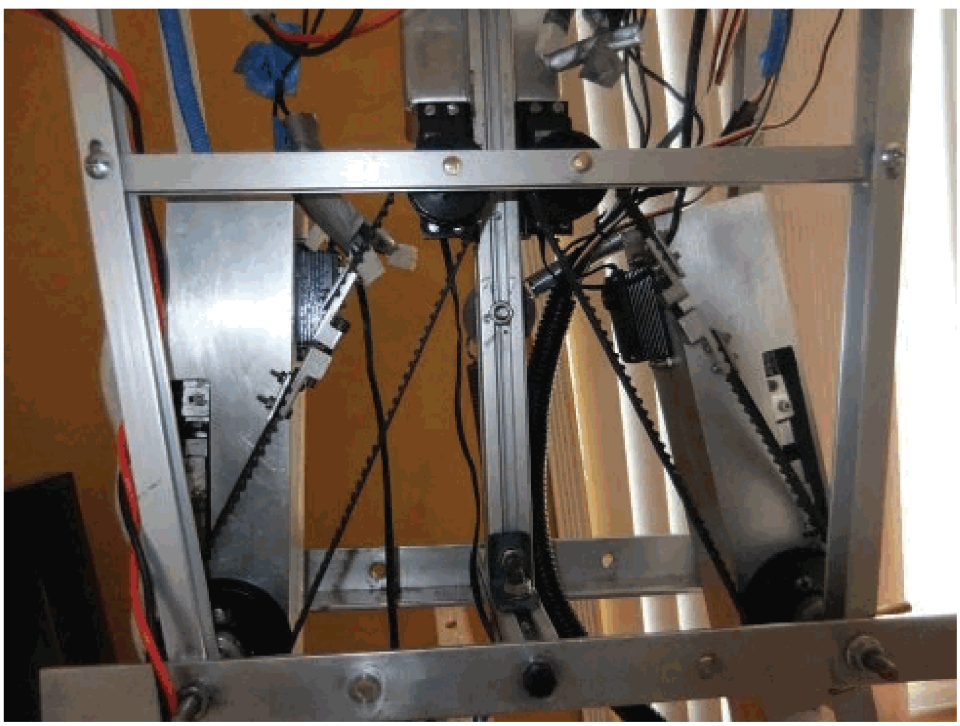
We removed the pushrods controlling the two axes of each foot and replaced them with a simple pulley and belt system.

We replaced the MXL80 belts in the system with larger XL belts and pulleys that are less likely to slip and skip. We designed a new belt tensioner system to allow easy tightening of the belts.

We put all the pulleys on .25 shafts. We now support those shafts to prevent the very tight belts from deforming the system.

All of the bushings have been replaced with .25 ID bearings to support heavy loads.

We also became more proficient with writing g-code to run the CNC mill. Now every cut, hole, and shaping of the



materials we fabricate are cut with precision and repeatability. We will be replacing most of the servos with ones that have much higher torque.

For very high torque areas, we will be increasing the pulley ratio. Some joints only require 45 degree movement. This allows a servo with 180 degree rotation to be geared down by a factor of four.

So, our new servos that are rated at about 1,600 oz-in should provide true 3,000 oz-in of torque.

The Math

Before we begin talking about the specifics involved in Watson, let's start with a quick refresher on units, levers and pulleys, and forces.

Units

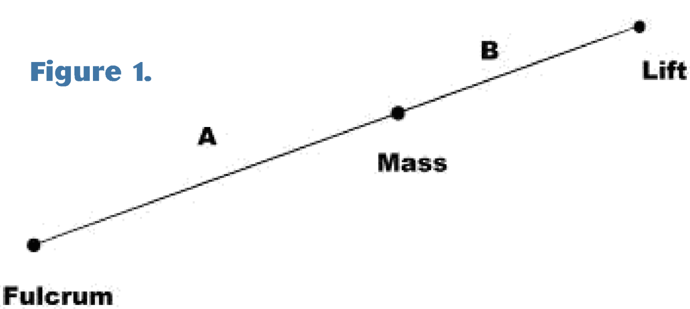
The fundamental units we need to work with are distance, time, and mass. In the SI system, these are respectively measured in meters, seconds, and kilograms. You may be more familiar with feet (0.3048 meters) and pounds (0.4536 Kg), but they are easily converted.

Derived from the fundamental units we have Speed (meters / seconds), Acceleration (meters / seconds^2), and Force (kg * meters / seconds^2). Force can be a bit confusing when it comes to acceleration due to gravity because we often just give the mass and assume it is on the surface of the Earth with its gravitational pull of 9.8 M/S^2. If our robot weighs 100 pounds here on Earth, it would only weigh about 16.6 pounds on the Moon, with a gravitational pull of only 1.622 M/S^2. In spite of weighing 16.6 pounds (of force) on the Moon, it would still have a mass of 100 pounds.

Levers

In the simple lever in **Figure 1**, the Fulcrum is fixed and at rest; a mass M is pulling down with a force of 9.8 * M. To keep everything balanced, we would need to lift the end with a force of (9.8 * M * A) / (A + B).

Figure 1.



For example, if we had a 10 pound weight in the exact center of a 2' board, we would need to lift with (10 * 1) / (1 + 1), or five pounds of force.

Torque

Torque is angular force rotating around a center point. Servos provide torque to move our robot, and their specs include how much torque they can exert. The units typically given are in either oz-in or kilogram centimeters.

For example, a Hitec HS-645MG servo provides 133 oz-in of torque. This means if you attach a one inch lever to this servo, it would be able to lift a 133 oz weight. In practice, the specs given are under optimal conditions while moving. For holding a static position (such as a standing robot), it is rare to get more than half the rated spec from any servo.

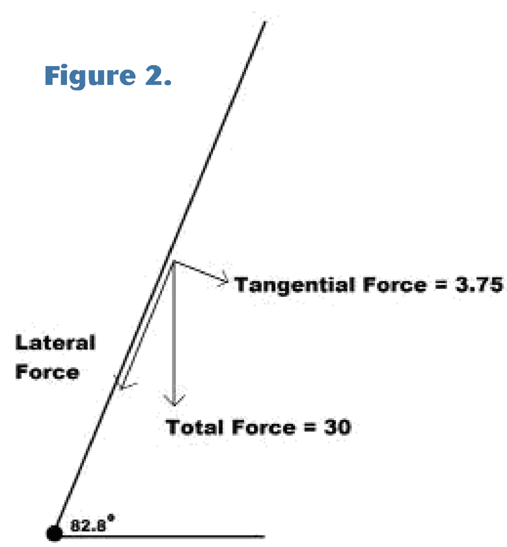
Now, let's do a simple calculation for a simplified robot, standing up. It has a single servo at the ankle, one foot, and a straight body. The mass is uniformly distributed along the body. To keep the center of gravity in the center of the foot for optimal stability, we need to lean forward a little bit. Let's give this robot the following parameters:

Height	Two meters
Body Mass	30 kg
Foot Length	.25 meters

First, we need the angle at the ankle joint. Because we want the center of mass in the middle of the foot, we conveniently get the top directly above the toe, giving us a nice right triangle.

We know Foot = Body * cos(angle), so angle = arccos(Foot/Body), or about 82.8 degrees. The lateral force is transferred through the rigid body to the ground, and the servo does not need to deal with that. It is only the

Figure 2.



tangential force we need to overcome, which is Mass * cos(82.8), or about 3.75 kg of force, at half the body height from the servo. This conveniently is 3.75 kilogram meters. (Refer to **Figure 2**.)

Multiply by 35.274 to convert kg to ounces and by 39.37 to convert meters to inches, and we can convert this to about 5,208 oz-in. Using the Hitec servo previously mentioned, we would need 79 of them just to hold this position. From this one simple calculation, we can see why 12" tall robots weighing only a few pounds are so popular!

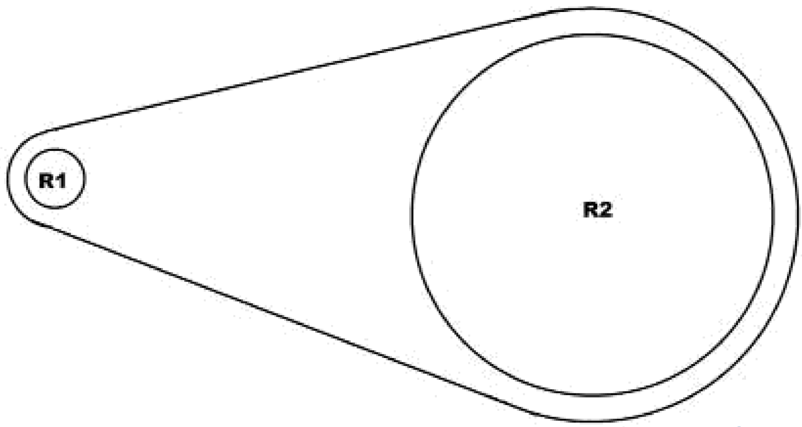


Figure 3.

Pulleys

Most servos just don't deliver that much torque, so we need to gear them down with pulleys.

Referring to **Figure 3**, if we have a servo on the left attached to a pulley with radius R1, turning the pulley on the right with radius R2, the shaft at the center of the second pulley will turn slower but with more torque. The speed will be reduced by a factor of R1/R2, but will increase the torque by R2/R1. For example, using the Hitec HS-645MG servo with a 1" pulley on the left and a 79" pulley on the right, we could make it work. (Let's ignore the fact that our simple robot happens to be about 79" tall.)

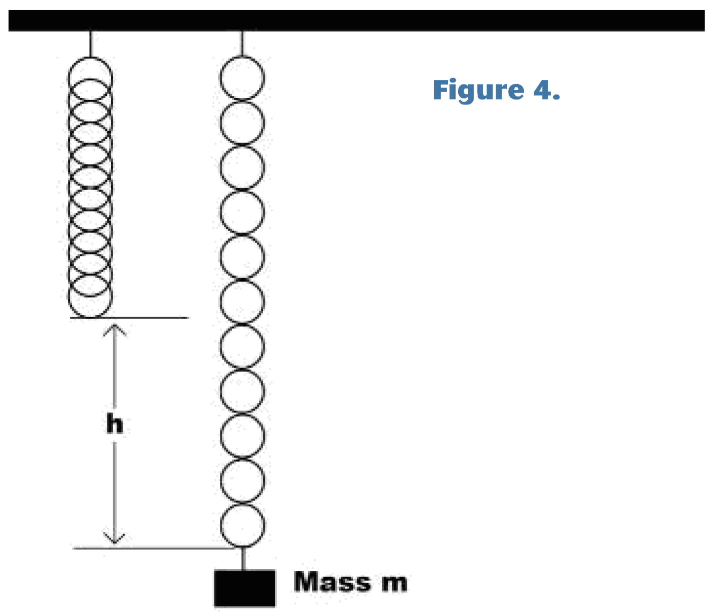


Figure 4.

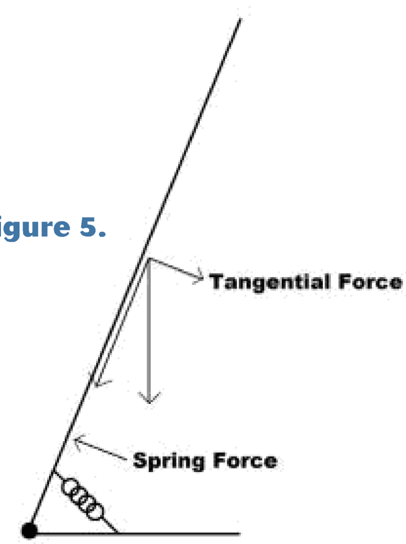
Springs

To further lessen the torque requirements of the servos, we can employ springs. When we attach a weight with mass *m* as in **Figure 4**, the spring stretches by the amount *h*. This relation is linear: Twice the weight will stretch twice as far.

In the previous example, we needed a 1:79 ratio pulley to overcome the torque on the ankle servo to stand up. Adding a simple spring allows us to offload most of the force from the servo.

Instead of the force as a function of lean angle being [Weight * cos(angle)], adding a spring makes it [Weight * cos(angle) - (K * sin(angle))]. By choosing the right spring constant K, we can now have the robot standing with little to no force needed from the servos. Take a look at **Figure 5**.

Figure 5.



Next time, we will present the last rebuild before the competition. It will include the new servos, and the improved feet and hips. We will also install the vision system for testing. **SV**

Making Better Arduino Robots with the ArdBot II

Part 3

By Gordon McComb

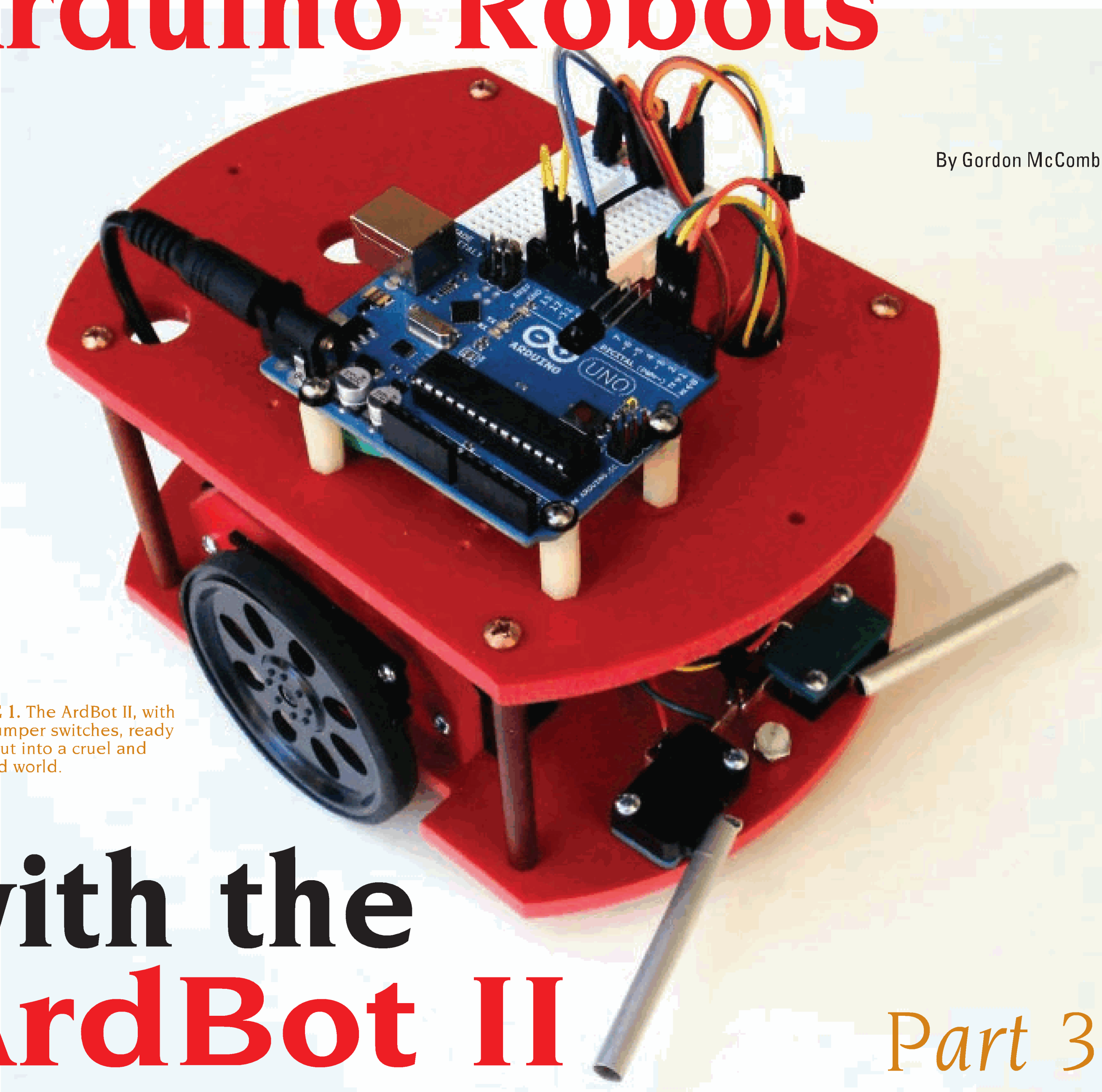


FIGURE 1. The ArdBot II, with front bumper switches, ready to set out into a cruel and crowded world.

Robots use sensors to know what's happening around them. Sensors don't have to be expensive or complicated to provide your bot with a rich overview of its world. A robot can perform a remarkable amount of work with just the sense of touch from a $2 switch.

This month, we'll learn about interfacing switches to your Arduino robot, along with how to use the information these sensors provide to interactively command a robot's motors. This is a critical and fundamental building block of most any autonomous robot. Once you learn how to use a switch, you can apply the same techniques to many other types of sensors, as well.

Take note: In the previous installments of this series, I mentioned certain accessories — like the front leaf switches and piezo speaker — are optional. In this article, I'll present some Arduino sketches that make use of these options. In order to enjoy the full benefit of the sketches presented here, you'll need to have these options installed on your ArdBot II.

Go to **www.servomagazine.com/index.php?/magazine/article/october2013_McComb** for any additional files and/or downloads associated with this article. You can also post comments.

Playing With Switches

Robots are at their best when they react to something. It can be light or touch or heat, or most any other sensation. Without the ability to detect its surroundings and then act on it, our bots are nothing more than those insidious bodiless brains that captured Captain Kirk, then put him in a fighting ring with a girl in silver shorts, and shouted "I wager 100 quatloos on the newcomers!"

To avoid having robots like that, we attach switches and other sensors, then write programming code to sense when those sensors are activated. That's exactly what we'll do with the ArdBot II's front switches.

Recall from Part 2 that our robot uses ordinary spring-loaded leaf switches; their contact area was increased with the use of rubber or thin-walled aluminum tubing. Refer to **Figure 1** for a photo of the prototype ArdBot II, and how the switches are oriented on the front to detect obstacles straight ahead.

Using a few of the Arduino's digital input/output (I/O) pins, we can connect these switches to the microcontroller and detect when the ArdBot II has made contact with an object.

Listing 1 shows a short sketch that demonstrates this process. The code makes use of the Arduino's external interrupt feature, whereby the two I/O pins used with the switches are constantly monitored by the microcontroller's hardware itself.

Using the Arduino's external hardware interrupt feature simplifies your code, and makes your robot more reactive. Let's consider the alternative method — called polling — before delving further into interrupts.

With polling, your sketch must repeatedly check the status of the two switches. If a switch is closed,

LISTING 1. *ardbot2_switches.*

```
#include <Servo.h>

Servo servoLeft;                 // Define left servo
Servo servoRight;                // Define right servo

volatile int pbLeft = LOW;      // Flag variable for left switch
volatile int pbRight = LOW;     // Flag variable for right switch
boolean started = false;        // True after robot first starts
moving

void setup() {
  // Set pin modes for switches
  pinMode(2, INPUT);
  pinMode(3, INPUT);
  pinMode(4, OUTPUT);

  // Set internal pull up resistors for switches
  digitalWrite(2, HIGH);     // Right switch input
  digitalWrite(3, HIGH);     // Left switch input

  // Use pin 4 as a ground connection
  digitalWrite(4, LOW);

  servoLeft.attach(10);     // Set left servo to digital pin 10
  servoRight.attach(9);     // Set right servo to digital pin 9

  Serial.begin(9600);       // Start serial for debugging

  // Set up interrupts
  attachInterrupt(0, hitRight, FALLING);
  attachInterrupt(1, hitLeft, FALLING);

  started = true;           // Okay to allow robot to start
moving
}

void loop() {

  if (pbLeft == HIGH) {              // If left bumper hit
    reverse();                        // Reverse for 1/2 second
    delay(500);
    spinRight();                      // Spin for 1.5 seconds
    delay(1500);
    forward();                        // Go forward again
    pbLeft = LOW;                     // Reset flag shows "hit"
    Serial.println("pbLeft");
  }

  if (pbRight == HIGH) {             // If right bumper hit
    reverse();
    delay(500);
    spinLeft();
    delay(1500);
    forward();
    pbRight = LOW;
    Serial.println("pbRight");
  }

}

// Motion routines for forward, reverse, turns, and stop
void forward() {
  servoLeft.write(180);
  servoRight.write(0);
}
void reverse() {
  servoLeft.write(0);
  servoRight.write(180);
}
void spinLeft() {
```

Continued

LISTING 1. *ardbot2_switches* continued.

```
   servoLeft.write(0);
   servoRight.write(0);
}
void spinRight() {
   servoLeft.write(180);
   servoRight.write(180);
}
void turnLeftFwd() {
   servoLeft.write(90);
   servoRight.write(0);
}
void turnRightFwd() {
   servoLeft.write(180);
   servoRight.write(90);
}
void turnLeftRev() {
   servoLeft.write(90);
   servoRight.write(180);
}
void turnRightRev() {
   servoLeft.write(0);
   servoRight.write(90);
}
void stopRobot() {
   servoLeft.write(90);
   servoRight.write(90);
}

// Interrupt handlers
void hitLeft() {
   if (started)                   // If robot has started set flag to high
     pbLeft = HIGH;
}
void hitRight() {
   if (started)                   // Same as left switch handler
     pbRight = HIGH;
}
```

Sources

Budget Robotics
Precut ArdBot II body chassis, with all assembly hardware
www.budgetrobotics.com

Selected sources for miniature leaf switches, piezo elements:

All Electronics
www.allelectronics.com

BG Micro
www.bgmicro.com

Jameco
www.jameco.com

Parallax
www.parallax.com

Pololu
www.pololu.com

the robot is commanded to steer to a new heading. The switches are checked — polled — many times each second.

Polling is an acceptable method when the sketch is relatively simple and the demands on the Arduino are light. For code that is more processing intensive, there is a chance the controller will miss when a leaf switch has closed. It'll be busy doing something else in between polls, and unaware anything has happened.

In truth, you can have a basic sketch and it will still detect 99 percent of all switch closures. The reason: The switch will likely be closed for what are very long periods of time to a microcontroller. As the sketch gets more complicated, the other code running on the Arduino is called upon to perform other tasks, so some switch closures may not be detected.

To ease the burden on the Arduino, you can use hardware interrupts where special code is run if — *and only when* — a specific external event occurs. In any Arduino sketch, the main body of the program is the *loop()* function. Programming instructions inside this function are repeated indefinitely.

With polling, the code to check the switches must be part of the *loop()* function, and repeats with each pass through the loop. When using hardware interrupts, there is no explicit code inside the *loop()* that monitors the switch inputs. That job is taken care of by an *interrupt handler* function. The code in the *handler(s)* only runs when the I/O pin associated with the interrupt is triggered.

The Arduino Uno supports two hardware interrupts (the Arduino Mega supports six) that are internally connected within the Arduino to digital pins D2 and D3. Of course, these are the pins that the leaf switches are connected to.

You will note a couple of interesting things about the sketch in **Listing 1**:

- The three I/O pins used with the switches — pins D2, D3, and D4 — are defined at the top of the *setup()* function. Pins D2 and D3 are made inputs, naturally, as they are used to determine when either switch has closed. Pin D4 is made an output and set LOW. Why? This permits the pin to act as a simple ground connection for the switches, and makes it easier to construct connectors for it. This method is acceptable when the sensor pulls only minimal current from the Arduino, which is the case with switches.
- The hardware interrupts are set up with the two lines:

```
attachInterrupt(0, hitRight, FALLING);
attachInterrupt(1, hitLeft, FALLING);
```

Note that the interrupts are referred to as *0* and *1*. These correspond to pins D2 and D3, respectively. The labels *hitLeft* and *hitRight* are the *handler* functions (see below) that are called when the interrupt is triggered.

Finally, *FALLING* is a built-in constant that tells the Arduino to trigger the interrupt on a HIGH-to-LOW signal transition. We look for a HIGH-to-LOW transition because when the switch is open, it's held HIGH — thanks to an internal pull-up resistor provided on the Arduino.

This resistor is electrically connected in-circuit because within the *setup()* function, we set the value of the pin as both an INPUT, then declared its value as HIGH.

Here is the process used to set the pull-up resistor on any pin defined as an INPUT:

- Two variables are used to keep track of the current state of the left and right bumper switches; *pbLeft* and *pbRight* are Boolean variables that store a true or false value. The value is true when the corresponding switch has just been activated. Note that these variables are defined at the top of the sketch using the *volatile* keyword. This is a special instruction for the Arduino compiler to treat the variables in a special way because we intend to use them inside interrupts. If we should forget to use this keyword, the sketch may behave erratically.
- The functions *hitLeft* and *hitRight* — located at the bottom of the sketch — are automatically executed whenever a corresponding interrupt is triggered. Note that the *interrupt handler* functions are not directly called in your *loop()* code. They are only executed when an interrupt occurs.

LISTING 2. *ardbot_sound.*

```
#include <Servo.h>
#define NOTE_B3  247
#define NOTE_C3  131
#define NOTE_C4  262
#define NOTE_D6  1175

Servo servoLeft;              // Define left servo
Servo servoRight;             // Define right servo

volatile int pbLeft = LOW;
volatile int pbRight = LOW;
boolean started = false;

void setup() {
  // Set pin modes for switches
  pinMode(2, INPUT);
  pinMode(3, INPUT);
  pinMode(4, OUTPUT);
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);     // Serves as ground connection

  pinMode(12, OUTPUT);      // Ground for speaker
  digitalWrite(12, LOW);

  servoLeft.attach(10);     // Set left servo to digital pin 10
  servoRight.attach(9);     // Set right servo to digital pin 9

  Serial.begin(9600);

  // Set up interrupts
  attachInterrupt(0, hitRight, FALLING);
  attachInterrupt(1, hitLeft, FALLING);

  started = true;
}

void loop() {

  if (pbLeft == HIGH) {           // If left bumper hit
    int tones[] = {NOTE_C4, NOTE_B3, NOTE_C4};
    int toneDurations[] = {4,4,4};
    reverse();
    makeTone(tones, toneDurations, sizeof(tones)/sizeof(int));
    delay(500);
    spinRight();
    delay(1500);
    forward();
    pbLeft = LOW;
    Serial.println("pbLeft");
  }

  if (pbRight == HIGH) {          // If right bumper hit
    int tones[] = {NOTE_D6, NOTE_C3};
    int toneDurations[] = {4,4};
    reverse();
    makeTone(tones, toneDurations, sizeof(tones)/sizeof(int));
    delay(500);
    spinLeft();
    delay(1500);
    forward();
    pbRight = LOW;
    Serial.println("pbRight");
  }

}

void makeTone(int tones[], int toneDurations[], int length) {
  // Iterate notes of tune
  for (int thisNote = 0; thisNote < length; thisNote++) {

    //Calculate the note duration
    int toneDuration = 1000/toneDurations[thisNote];
```

Continued

LISTING 2. *ardbot_sound* continued.

```
     tone(13, tones[thisNote],toneDuration);

     //Add slight pause between notes
     int pauseBetweenNotes = toneDuration * 1.30;
     delay(toneDuration * 1.30);
     noTone(13);                        // Stop tone
   }
   return;
}

// From here down copy from Listing 1
// Motion routines and Interrupt handlers
```

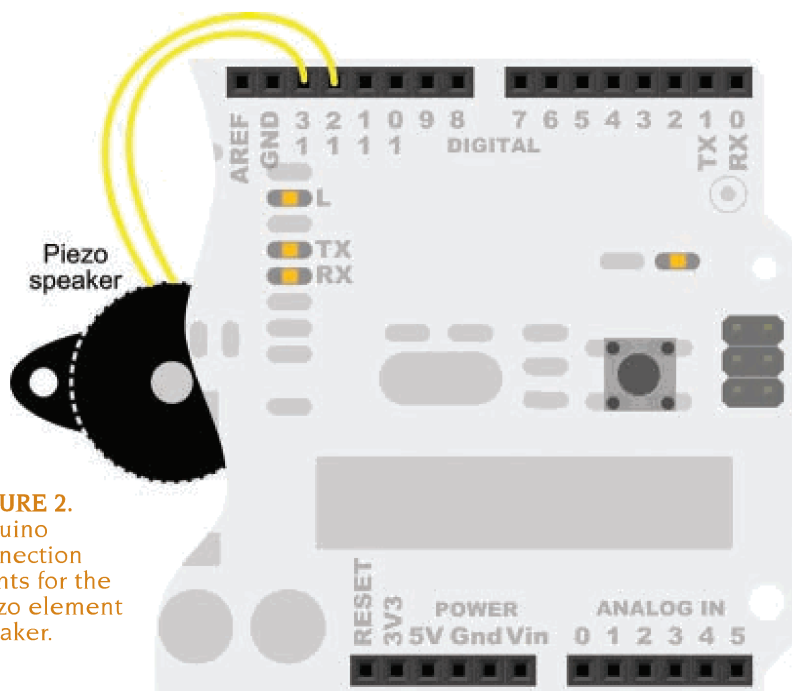


FIGURE 2. Arduino connection points for the piezo element speaker.

You might be wondering why the code to control the servos isn't in the interrupt handlers. The reason is this: The *delay* statement — which is used to steer the robot around an obstacle — is disabled while processing an interrupt. Our interrupt code merely sets a "flag" variable that indicates its corresponding switch has been triggered. This flag is acted upon the next time the *loop()* code is repeated. It's also considered better programming practice to not place time-intensive functionality within interrupt handlers.

The code to steer the bot in a new direction when it hits something is handled in the *loop()* function inside two *if* statements. If either the *pbLeft* or *pbRight* flag variables have been set, the robot backs up, does a quick about-face spin turn, then heads off in a new direction.

Before continuing, the flag variable is reset for the next time. In case you want to monitor the activity of your ArdBot II in the serial monitor window, the code includes some *Serial.println* statements to serve in simple debugging.

Adding Sound Feedback

Feedback is when your robot tells you something about what it's doing. One form of feedback is the *Serial.println* debugging statements found in the **Listing 1** sketch. If the robot isn't behaving as it should, you can open the serial monitor window and check that those debugging statements are appearing. If you see the text but the bot isn't moving the way it should, that tells you the basic switch logic is working because that code is being executed. So, that means the problem is probably in the servos or the servo power.

You can't always have your robot connected to your PC so that you can view debugging information. The Arduino's LED (pre-wired to pin D13) is one way to provide quick and easy feedback to keep you updated on your robot's current status. It's not always convenient to view the LED, however, and unless you learn Morse Code the variety of messages your bot can pass on to you is limited.

Another way is through sound. By connecting a small piezo speaker directly to the Arduino, you can produce various chirps, beeps, hums, and other effects that can be used to indicate function. The sound may not be extremely loud, but that's not a problem. You only need to be able to hear it when you're in close proximity to your ArdBot II, checking to make sure everything is working as it should.

Refer back to Part 2 for details on adding a piezo speaker to your Arduino. **Figure 2** provides a reminder of the connections. Since the piezo element doesn't need an amplifier (at least we're not using one with the ArdBot II), hookup is quick and easy — just two wires to the Arduino's D12 and D13 I/O pins.

Important! As noted in Part 2 of this series, you need to use a piezo speaker, not a standard dynamic speaker — one that has a magnet and voice coil. Those kinds of speakers can draw too much current from the Arduino's output pins. Also, be sure to use a piezo speaker (capable of producing multiple tones) and not a piezo buzzer.

Listing 2 demonstrates playing short tones through the piezo speaker in response to collisions detected by the robot's two switches. A different short ditty plays depending on which switch is struck. Because the piezo

speaker is driven through pin D13, the Arduino's built-in LED will also momentarily flash during sound annunciation. This serves as an additional visual cue that the robot is communicating to you.

For tune making, I'm using a variation of example code provided on the *arduino.cc* website for playing melodies. I've simplified it a bit, and the routines are encapsulated in a simple function labeled *makeTone*. This function is called using three parameters:

- *tones* — an array of frequencies to play.
- *toneDurations* — another array that indicates the number of "beats" for each note.
- The number of *int* elements in the *tones* array. This allows the *makeTone* function to know how many notes to play. To play all the notes — the usual procedure — I use some well-known code that self-describes the number of elements in the array. Depending on your usage, you can alter the number of tones to play to be less than the actual number of elements in the *tones* array.

Here is an example of setting up the tones and duration, and then calling the *makeTone* function. This is for when the left bumper switch has been hit:

```
int tones[] = {NOTE_C4, NOTE_B3, NOTE_C4};
int toneDurations[] = {4,4,4};
reverse();
makeTone(tones, toneDurations,
sizeof(tones)/sizeof(int));
```

(The process is the same for the right bumper, except for that switch I've used two instead of three notes, and the notes are different. This is to differentiate between a right and left bumper strike.)

Notice that rather than actual tone frequencies, the tones are defined as constants. This makes it easier to refer to them in code. The constants are defined at the top of the sketch:

```
#define NOTE_B3   247
#define NOTE_C3   131
```

and so on. Also notice these are compiler *#define* definitions, which means they sort-of look like variables but they aren't actual variables. They don't take up any of the Arduino's memory. Rather, when the sketch is compiled, the Arduino IDE (integrated development environment) software substitutes the constant name — such as *NOTE_B3* — which is its actual value; in this case, 247. That number corresponds to a frequency of 247 Hz (cycles per second).

As a point of reference, a concert A pitch is denoted as *NOTE_A4*. It's the A right below middle C on a piano keyboard. This pitch has a (more or less) standardized frequency of 440 Hz.

You can experiment with more tones using **Listing 3**. It's the same as **Listing 2** but rather than coding the pitches in the main sketch, it uses an external file called *pitches.h*. This file is included at the article link for all the sketches for this installment.

To alter a tone, just copy its constant name (NOTE_F7, or NOTE_D6) and paste it into your sketch. Be sure to place the *pitches.h* file in the same folder as your **Listing 3** (*ardbot_sound_pitches.ino*) sketch. Otherwise, the Arduino IDE won't be able to find it, and will display a series of errors when it cannot fathom what you mean by "NOTE_F7."

LISTING 3. *ardbot_sound_pitches.*

```
#include <Servo.h>
#include "pitches.h"

Servo servoLeft;                // Define left servo
Servo servoRight;               // Define right servo

// etc.
// Rest of Listing 2 is the same
```

Coming Up: Remote Control ArdBot

We've run out of room for this installment. I was going to introduce you to using an infrared sensor to operate your ArdBot II with a universal remote control, but alas, it'll have to wait until next time. In upcoming parts, you'll also discover some additional cool ways to allow your ArdBot II to think and act on its own. **SV**

About the Author

Gordon McComb is the author of the best-selling *Robot Builder's Bonanza* and the new *Arduino Robot Bonanza*, both published by McGraw-Hill.

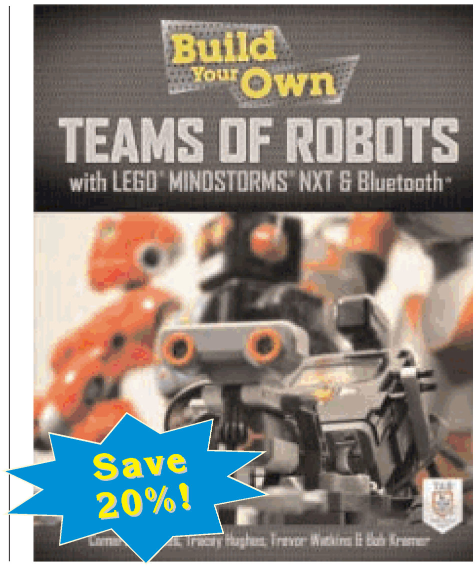
Refer to Part 1 of this series for a full list of mechanical parts for the ArdBot II.

The SERVO Webstore

Attention Subscribers ask about your discount on prices marked with an *

CD-ROM SPECIALS

10.2013 FEATURED

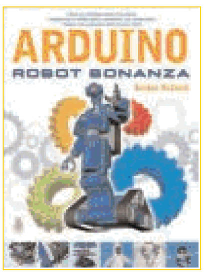


ROBOTICS

NEW!

Arduino Robot Bonanza

by Gordon McComb



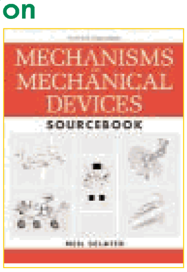
Create high-tech walking, talking, and thinking robots. "McComb hasn't missed a beat. It's an absolute winner!"

Breathe life into the robots of your dreams — without advanced electronics or programming skills. *Arduino Robot Bonanza* shows you how to build autonomous robots using ordinary tools and common parts. Learn how to wire things up, program your robot's brain, and add your own unique flair.

$29.95*

Mechanisms and Mechanical Devices Sourcebook 5th Edition

by Neil Sclater



Fully revised throughout, this abundantly illustrated reference describes proven mechanisms and mechanical devices. Each illustration represents a design concept that can easily be recycled for use in new or modified mechanical, electromechanical, or mechatronic products. Tutorials on the basics of mechanisms and motion control systems introduce you to those subjects or act as a refresher.

Reg $89.95

Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists

by Dustyn Roberts

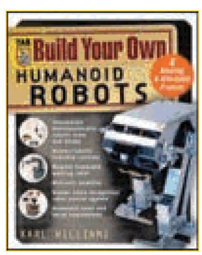
In *Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists*, you'll learn how to successfully build moving mechanisms through non-technical explanations, examples, and do-it-yourself projects — from kinetic art installations to creative toys to energy-harvesting devices. Photographs, illustrations, screenshots, and images of 3D models are included for each project.

$29.95*

Build Your Own Humanoid Robots

by Karl Williams

GREAT 'DROIDS, INDEED!

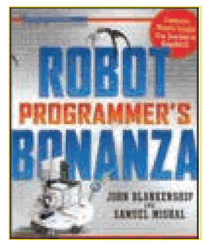


This unique guide to sophisticated robotics projects brings humanoid robot construction home to the hobbyist. Written by a well-known figure in the robotics community, *Build Your Own Humanoid Robots* provides step-by-step directions for six exciting projects, each costing less than $300. Together, they form the essential ingredients for making your own humanoid robot.

$24.95*

Robot Programmer's Bonanza

by John Blankenship, Samuel Mishal



The first hands-on programming guide for today's robot hobbyist!

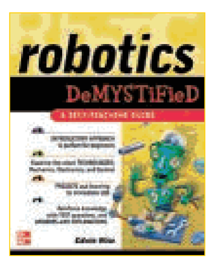
Get ready to reach into your programming toolbox and control a robot like never before! *Robot Programmer's Bonanza* is the one-stop guide for everyone from robot novices to advanced hobbyists who are ready to go beyond just building robots and start programming them to perform useful tasks.

$29.95

Robotics Demystified

by Edwin Wise

YOU DON'T NEED ARTIFICIAL INTELLIGENCE TO LEARN ROBOTICS!

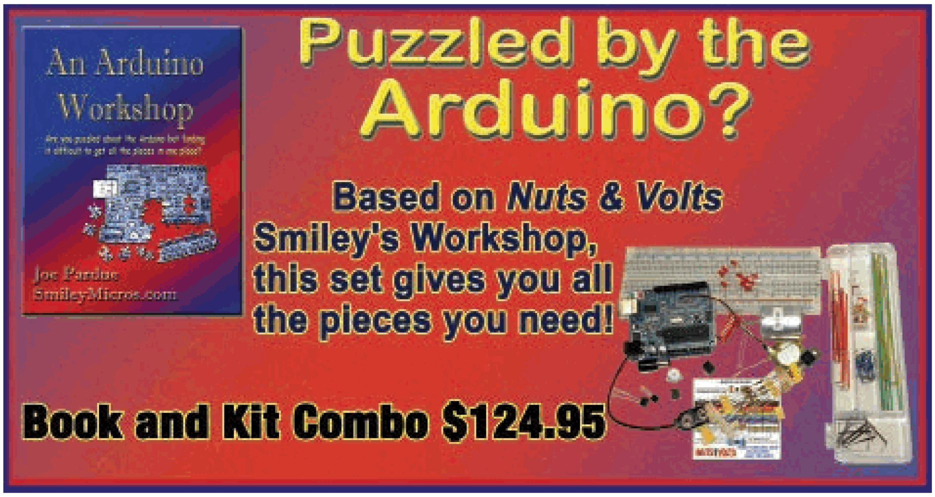


Now anyone with an interest in robotics can gain a deeper understanding — without formal training, unlimited time, or a genius IQ. In *Robotics Demystified*, expert robot builder and author Edwin Wise provides an effective and totally painless way to learn about the technologies used to build robots! **$19.95**

We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

Or order online
www.servomagazine.com

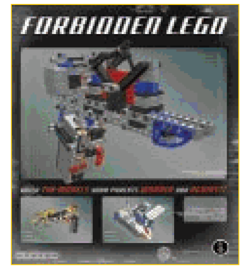
SPECIAL OFFERS



Forbidden LEGO

by Ulrik Pilegaard / Mike Dooley

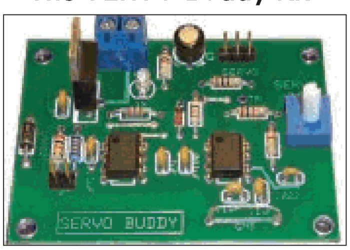
Forbidden LEGO introduces you to the type of free-style building that LEGO's master builders do for fun in the back room. Using LEGO bricks in combination with common household materials (from rubber bands and glue to plastic spoons and ping-pong balls) along with some very unorthodox building techniques, you'll learn to create working models that LEGO would never endorse.



Reg $24.95 Sale Price $19.95

PROJECTS

The SERVO Buddy Kit



An inexpensive circuit you can build to control a servo without a microcontroller.

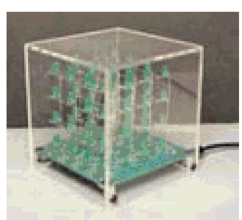
For more information, please check out the May 2008 issue or go to the SERVO webstore.

Includes an article reprint.

Subscriber's Price **$39.55**
Non-Subscriber's Price **$43.95**

3D LED Cube Kit

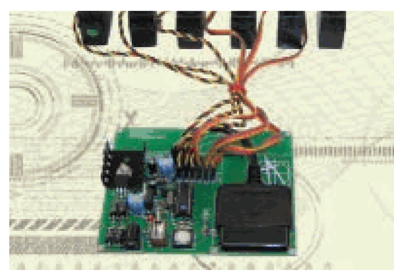
From the article "Build the 3D LED Matrix Cube" as seen in the August 2011 issue of *Nuts & Volts Magazine*.



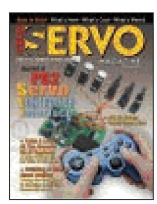
This kit shows you how to build a really cool 3D cube with a 4 x 4 x 4 monochromatic LED matrix which has a total of 64 LEDs. The preprogrammed microcontroller that includes 29 patterns that will automatically play with a runtime of approximately 6-1/2 minutes.
Colors available: Green, Red, Yellow & Blue.
Jig and plastic cases also available.

Subscriber's Price **$57.95**
Non-Subscriber's Price **$59.95**

PS2 Servomotor Controller Kit



This kit accompanied with your own PlayStation controller will allow you to control up to six servomotors. Includes all components and instruction manual.



For more information, please see the February 2011 edition of *SERVO Magazine*. Assembled units available!

Subscriber's Price
$79.95
Non-Subscriber's Price
$84.95

Twin brothers hack whatever's put in front of them, then tell you about it.

by Bryce Woolley and Evan Woolley

Go to **www.servomagazine.com/index.php?/magazine/article/october2013_TwinTweaks** to comment on this article.

Material Bot

ALUMINUM — A GREAT GENERAL-PURPOSE METAL.

Last time, we demonstrated that even an unassuming appliance like a printer can be turned into a fun fighting robot. Just as most anything can be turned into a robot, most anyone can make a robot with the right mix of confidence and knowledge.

An important and perhaps sometimes neglected building block of robotics knowledge is familiarity with basic materials. We've drawn upon our knowledge and experience with materials in every robotics project we've worked on, and knowing the difference between 6061 and 7075 has not only made us sound like nerds, but it has also improved our designs and enriched our robotics experience.

To help those looking to increase their fluency in robotics fundamentals, provide a useful refresher, or simply prepare for that inevitable *Jeopardy!* category on aluminum alloys, this month we'll provide an overview of some of our favorite building materials and where to find them.

Alloyed Forces

You can build a robot out of pretty much anything, but if you find yourself debating between general categories like wood or plastic or metal, this is the article for you. Having a more granular understanding of the materials available to you helps you make smarter design decisions and better robots.

Perhaps it's due to our history in combat robotics, but our preferred category of material from which to make a robot is metal. Far from a monolithic bore, the metal family is varied, exciting, and perhaps a little intimidating to the uninitiated.

A metal robot sounds cool, but what do I use? Isn't aluminum for soda cans and wrapping up baked goods? If it makes for good robot boxers, should I go with steel? Or something more exotic, like titanium?

These questions and more might make the initial selection of a material seem overwhelming, but a few guiding principles can help you make a smart selection that's right for your project.

With many robotics projects, two concerns always seem to loom larger than the others — weight and cost. Just like a dedicated Biggest Loser contestant, most competitive robots must fastidiously keep their weight down to acceptable levels. Most projects are also on a tight budget, making cost an important factor.

At first blush, these concerns may seem to rule out metallic designs in some circumstances. Plastic is lighter and cheaper, right?

Perhaps, but those are not necessarily the only questions you should be asking. For a robotics project, a better property to look at than weight alone is the strength to weight ratio. A piece of plastic may weigh less than the same size piece of metal, but the metal piece will have a better strength to weight ratio.

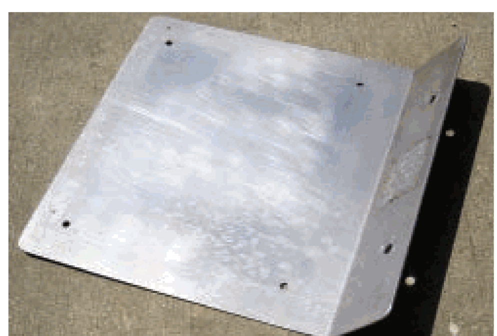
With that better ratio, it means you can use a thinner piece of aluminum rather than a piece of plastic for your robot floor without sacrificing strength. By using less material, you save on costs because most materials are priced by weight.

Other guiding questions that will help you select the best material for your robotics application are machinability, strength, and cool factor to name a few.

These questions can't be asked in a vacuum — the answers will depend on what part of the project you're using the material for.

With these questions in mind, we are ready to take a look at some of our favorite materials that we've used in various robotics projects.

First and foremost, our favorite material to work with is aluminum. Sure, they make soda cans and foil out of it, but if it's good enough for



ALUMINUM — IT'S SO BENDABLE.

IT'S DANGEROUS TO GO ALONE, SO TAKE THIS 7075-T6 ALUMINUM WITH YOU.

airplanes, it should be good enough for robots. Don't think that you can learn all you need to know simply by looking up lucky atomic number 13 on the periodic table, however. For a robotics project, you'll surely want to use an aluminum alloy.

An alloy is a mixture of elements. In an aluminum alloy, aluminum is the predominant element, and the additional alloying elements are usually zinc, magnesium, copper, or silicon to name a few. To help keep all of the different alloys straight, there is a system called the International Alloy Designation System. This system identifies alloys by a four-digit number, with the first number identifying the predominant alloying element.

Our two favorite aluminum alloys to use are 6061 and 7075; 6061 is part of the 6000 series of aluminum alloys, where magnesium and silicon are the primary alloying elements. 6061 is a widely used alloy of aluminum because it is easy to machine, easy to bend, and it has a decent strength to weight ratio. 6061 was often our preferred aluminum to use in conjunction with our FIRST robots, and it made appearances on every incarnation of Team 1079's MO.

We used 6061 on everything from MO's aluminum skirts (for the 2003 FIRST game Stack Attack) to an articulating arm on DreiMO (for the 2005 FIRST game Triple Play). 6061 was a better choice than other alloys because — in particular — 6061 is easy to weld, which is how we constructed DreiMO's articulating arm.

6061 is a fine choice as a general-purpose aluminum, but we'll always have a special place in our hearts for good old 7075. 7075 is part of the 7000 series, where zinc is the primary alloying element. 7075 aluminum has higher strength than other aluminum alloys, and it was often our preferred material for combat robot applications.

Our first combat robot Troublemaker sported walls made from 7075, and while some fearsome opponents were able to scrape it up a bit, none of them got through to the delicate electronics inside.

To be precise, our preferred type of 7075 aluminum is 7075-T6. While it sounds like the designation for a robot assassin from the future, T6 refers to the heat treatment of the material. Specifically, T6 refers to an alloy that has solution heat treatment and artificial aging. For purposes of combat roboteers, that means that 7075-T6 is perfect for sturdy armor and fearsome weapons.

The major downside to 7075 is that the machinability is lower than 6061. That means you'll have to work harder to drill holes in the metal, and shaping it by bending is often out of the question.

Once you've made the intellectual selection of the type of aluminum alloy you want to employ, there is still the practical concern of where to actually acquire it. Our favored source for keeping Robot Central stocked was our local recycling center, which had tons of wonderful scrap available on the cheap.

The proper alloys are easily identified by having one edge painted according to the alloy. 7075, for example, has one edge painted black. A local recycling center or scrapyard is bound to have plates, cylinders, and tubes of all shapes and sizes. If there isn't such a wonderful place nearby, you can always order raw materials from an online source like trusty McMaster-Carr.

Real Deal Steel

Since we're in the realm of alloys, it's a good time to talk about another popular one — steel. Steel is an alloy of iron, with the predominant alloying element often being carbon. While steel sounds like the perfect material for something like giant boxing robots, it's not something we have used on our bots all that often. The main reason is because steel is usually much heavier than aluminum, and a tough alloy like 7075 will mean that aluminum gives you all of the strength you need but not all of the weight you don't.

What we have often used steel for in Robot Central is to make things like fixtures to aid in the machining process. If you have an oddly shaped part that you need to put into a press, you can rig up a bracket out of a large piece of steel to provide you with a heavy, sturdy piece that won't be inclined to move around as you use your hands to work the machine.

For this sort of task, an angle piece of a steel alloy like 1018 is ideal. 1018 is a very common grade of carbon steel, sure to be found in abundance at your local scrapyard or recycling center, and can be obtained cheaply for a price of around 55 cents a pound or so.

There have been occasions where we've used a specific alloy of steel

RUN-OF-THE-MILL STEEL.

called chromoly. Chromoly takes its name from the predominant alloying elements of chromium and molybdenum, and it is stronger and harder than many other steel alloys. We used chromoly for weapons on our combat robots — for front mounted spikes on 60 lb Troublemaker and for twin axe blades on 30 lb Twibill Trouble.

CHROMOLY, BEFORE HEAT TREATMENT.

One of the fun aspects of working with chromoly is that it is a great candidate for heat treating, which is something you can do in your garage (with the proper safety precautions, of course).

To harden the spikes and blades, we used a heat treating process known as quenching. As the name suggests, quenching involves heating up the metal and then quickly submerging it in a liquid to cool it down. For our purposes, we heated up the metal with an acetylene torch and then quenched it in a motor oil bath. Water can also be used as a quenching medium, and while it can help achieve the best hardness (because the quenching is so fast), it can also cause distortions and cracks in the material.

A tip for do-it-yourself quenching: Rremember to use something like a metal tray to contain the quenching bath. If you use a plastic tub, your heated metal piece can melt right through the bottom of the tub like Walter White's first attempt at covering his criminal tracks.

Chromoly is a bit more exotic than mundane 1018 steel, and you may not be able to find it at just any old scrapyard. You can, however, order it online — we acquired raw material for our combat robot weapons again from trusty McMaster-Carr. Our final stop in steel country is a visit with spring steel. Spring steel is a steel alloy with a very high yield strength and with the ability to return to its original shape after significant bending. Spring steel is sometimes used by ambitious combat roboticists for thin sheets of armor that can take a lot of abuse even at low thicknesses.

We haven't used too much spring steel for one main reason: It is very difficult to work with. The same strength that makes it great armor means that it will be similarly impervious to your drill bits and saws. It's certainly a feasible material to work with for those that have the requisite patience.

Perhaps the most interesting part about the spring steel we've used is where we got it — it was a gift from our former FIRST mentor and *SERVO Magazine* technical editor Dan Danknick, as leftovers from his illustrious combat robotics days. It just goes to show that you should always be on the lookout for cool materials, and if you're in the hunt for something unique, remember to ask your friends!

Wrath of the Titanium

Titanium is a fun material to use, even though much of that might

SPRING STEEL — FOR THOSE WITH SAINTLY PATIENCE.

SOFT ALUMINUM SHEETING.

TITANIUM — THE JAGGED EDGE FROM THE PLASMA CUTTER LETS YOU KNOW IT DIDN'T GO QUIETLY ...

come from the perceived novelty of using something that has entered the popular lexicon to mean something super strong. Like aluminum, titanium is often alloyed with other metals to improve its properties, and titanium and its alloys display impressive strength and hardness without tipping the weight scales.

Titanium is as strong as you would expect for something named after the formidable Titans of Greek mythology, and it is a bear to machine. Our main project that we used titanium on was our 30 lb Twibill Trouble, where it was used to make the mounting brackets for the bot's two chromoly axe blades.

The titanium we used on Twibill Trouble was a quarter of an inch thick, and it would take a long time and some serious blades to cut it with a saw. Luckily, we had something far more awesome at our disposal — a plasma cutter. A plasma cutter works by passing an inert gas at high speed through an electrical arc. This creates plasma, which is hot enough to cut through even the most stubborn metals. The high speed jet of gas also works to blow away the molten metal.

U CHANNEL.

EMT, BUT NOT THE KIND FROM AN AMBULANCE

We didn't rely on exotic tools for all of our machining needs with the titanium, however. We still had plenty of holes to drill for mounting and to get the weight down for our combat robot, but there were some useful techniques we could employ to ensure that accomplishing our goals was as painless as possible.

Firstly — when machining something as hard as titanium — it is a good idea to invest in some special drill bits designed to work on hard materials. Using regular run-of-the-mill drill bits is bound to be a tedious and unpleasant experience. Softer drill bits will be slow, they'll wear out quickly, and if you're not careful you run the risk of work hardening the material and making it even tougher to drill through.

In addition to the right drill bit, having the right cutting fluid can make a difference too. When working with aluminum alloys like 7075, it is always a good idea to use something like WD-40 as a cutting fluid to avoid work hardening and too much heat buildup.

A special material like titanium calls for a special cutting fluid, and the one we used was a molybdenum-based cutting fluid whimsically referred to as Moly-D (and more technically referred to molybdenum disilicide).

Our experience with titanium also shows that you can get materials in far more interesting ways than a mundane trip to the scrapyard. The titanium we used on Twibill Trouble — as with much Western titanium — was Russian titanium.

In a deal that sounded like something straight out of a John le Carre novel, we obtained the Russian titanium not by heading to the scrapyard or ordering online, but rather by bartering for it and trading some high performance electric motors (the kind we use in our combat robots) for a couple of plates.

www.mcmaster.com

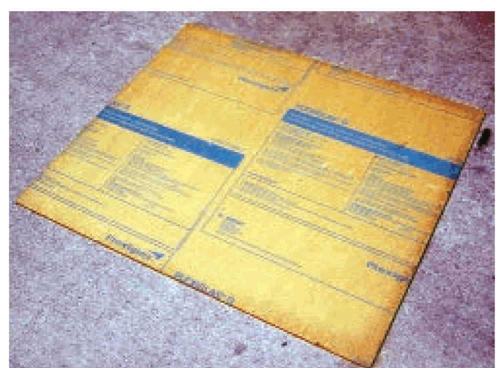
Materialism

There are a few other materials that we've used on many a project that should be a part of any roboticist's repertoire. While we fawned over 6061 and 7075 above, there are many other types of aluminum that are useful for things other than combat robot armor or weapons.

Softer alloys like those in the 5000 series (with the predominant alloying element magnesium) can often be found at your local home improvement store in thin sheets, usually rolled up. Soft thin aluminum sheets are great for applications requiring a lot of bending and shaping.

One instance where we used this kind of aluminum was on our FIRST robots to cover our drive train chains. The soft aluminum could easily be shaped into a guard, and shaped aluminum has a much higher cool factor than something like plastic netting (though we have used that for similar purposes).

Something else to know about the aluminum at your disposal for robotics projects is that it comes in a variety of shapes and sizes. Extruded aluminum can come in tubes, U shapes, plus shapes, and many more.



ACRYLIC — GOOD FOR RAPID PROTOTYPING (NOT FOR STOPPING BULLETS).

PVC — NOT JUST FOR HOME IMPROVEMENT.

Extruded aluminum like this is great for framing up robots or for designing small brackets where the geometry gives it more strength than a flat plate.

Just as with aluminum, steel comes in many flavors. One variety in particular that we've often used is called EMT, which stands for Electrical Metallic Tubing. EMT is easily found at your local hardware or home improvement store because it is generally used as an electrical conduit. EMT is also a good fit for robotics projects because it is easy to find, cheap, thin-walled, and lightweight, and reasonably sturdy.

It can be bent pretty easily (even without the services of a Bending Unit 22) and can be used to rapidly prototype designs by making it simple to frame something up. EMT has made some appearances in some of our final robot designs too; most notably as the frame for the hopper on DreiMO — Team 1079's FIRST robot for the 2006 game Aim High. We needed a light frame that could withstand the hubbub of an energetic FIRST match, and the EMT served us well.

Aluminum (and sometimes steel) are also seen in the popular incarnation known as diamond plate — instantly recognizable because of its distinct diamond pattern. The diamond pattern is often used in automotive applications because the raised diamonds prevent those walking on it from slipping, but the pattern also serves a useful structural purpose. The diamond pattern increases the strength of the material and its resistance to bending. It also adds a nice little flair of cool factor.

Another useful material is acrylic glass — a shatterproof alternative to glass that is useful for rapid prototyping (and is often known by its trade name Plexiglass). Acrylic sheets featured prominently in Evan's mechanical engineering curriculum at UCSD, where it was often used in conjunction with a laser cutter to rapidly bring designs from the page to the real world.

Acrylic is also familiar to FIRST teams the world over as a frequently used protective guard that keeps delicate electronics visible but protected from the cruelties of the world.

Similar to EMT, PVC pipe is another common home improvement store find that can serve you well in a robotics context. Polyvinyl Chloride is a very common type of plastic, and its common usage in plumbing means that any hardware store will have a plethora of elbows, t-joints, and end caps that can make the world feel like one big Erector set.

All of these premade parts make PVC an ideal candidate for a robot frame, which is exactly what we did with our entry into the aquatic 2008 MATE ROV Competition (check out our August and September 2008 articles for more on that underwater adventure).

Finally, we encourage all roboticists to look at materials as not just an obligatory design decision, but as a source of inspiration. If you find a material that speaks to you like you're in a Pier 1 Imports commercial, by all means run with it. We've based entire robot designs on materials we thought were cool and wanted to incorporate simply for the sake of cool factor.

The design of our combat robots was largely dictated by our desire to use carbon fiber plenum tops in the design. The plenum tops were part of racecar air intake manifolds — something we always delighted in telling curious passers-by because it was so cool.

That's what having a good background in materials should do in addition to helping you make smart design decisions — it should help you feel excitement and pride in your project, even with the aspects that seem so basic like what kind of metal to use. SV

DIAMOND PLATE ALUMINUM IS A ROBOT'S BEST FRIEND.

CARBON FIBER PLENUM TOP, FOR THE COOL FACTOR, OF COURSE!

Special Thanks To

Dan Danknick — for the cool materials and even cooler mentorship.

by Tom Carroll

Go to **www.servomagazine.com/index.php?/magazine/article/october2013_ThenNow** to post comments on this article.

The LEGO MINDSTORMS Evolution — RCX to EV3

When many of us think about LEGO, we recall all those colorful plastic blocks that kids have strewn about the house, or the time we unexpectedly found one or more stuck to our foot in the middle of the night. If you or your kids have ever worked with the LEGO MINDSTORMS sets to learn robotics, you may have already seen LEGO's latest MINDSTORMS robot kit: the EV3 system. Let's take a look into the historical background of MINDSTORMS and compare the three newest versions, with a direct comparison of the new EV3 and the MINDSTORMS NXT that was released back in 2006.

YouTube videos from the Consumer Electronics Show held in Las Vegas, NV this past January highlighted both versions of the new LEGO MINDSTORMS® EV3 robot: one available for home use and one available for use in the classroom. One video showed a LEGO robot rover that was able to track lines on the floor. It could be fitted with one of four different interchangeable tools such as a gripper, a hammer, a ball launcher, and even a spinning chopper. **Figure 1** shows one of the new EV3 tracked robots.

LEGO also demonstrated a robot snake that slithered about on sets of wheels and could snap at people when onboard IR sensors detected the warmth of their hand. Another version that was available in the kit was a scorpion that could hunt down and track a user's IR controller, and use its tail to blast the controller with small plastic balls. There is also a treaded LEGO robot with a powerful gripper that can carry something as heavy as a soda can.

Many *SERVO* readers began their interest in robotics by constructing and programming LEGO robots, and later stepped up to more complex robots as their fascination with robots progressed. This is exactly what LEGO hopes to accomplish with their excellent educational systems.

The MINDSTORMS product line is aimed primarily at children around the age of eight. However, many older kids love playing with the different MINDSTORMS sets — myself included.

Building these things is addictive to all ages. It is this type of robot that first interests young minds and allows them to progress upward to FIRST LEGO League robot competitions and later to FIRST robot competitions in high school.

A lot of folks construct robots in both types of kits for fun, but the knowledge that anyone can gain from building these robots is directly applicable to more advanced robotic projects. LEGO has developed a thorough STEM (Science, Technology, Engineering, Math) curriculum to use with their MINDSTORMS systems in school classrooms.

A Little LEGO History

The LEGO MINDSTORMS series of educational robot kits has been

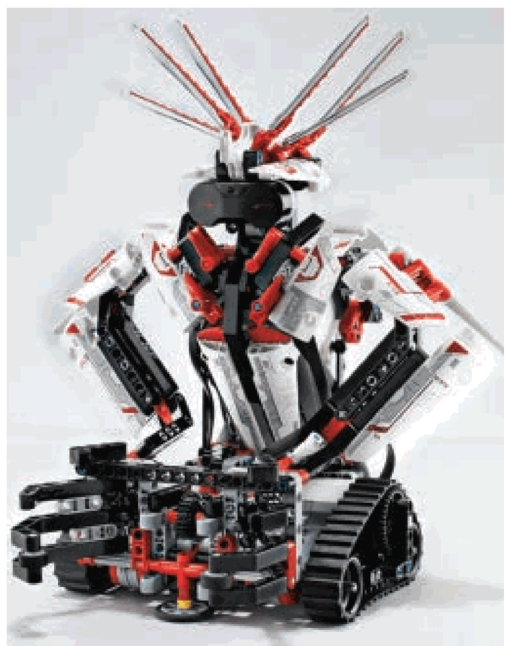


Figure 1. New LEGO EV3 tracked robot from the home version kit.

extremely popular with children, as well as robotics hobbyists for almost two decades now. The snap-together plastic blocks were a mainstay of kid's toy boxes for many years, and are still a best seller.

Becoming the world's best-known children's toy blocks/bricks was not a particularly easy rise to fame. The basis for the line of toy bricks began eight decades ago.

The LEGO Group was started by Ole' Kirk Christiansen — a carpenter from Billund, Denmark who began making wooden toys in 1932. In 1934, his company came to be called "lego," from the Danish phrase *leg godt* which means "play well." LEGO began manufacturing interlocking toy bricks in 1949.

A Chinese company started to market similar bricks in 2002, and was sued by LEGO and lost. A German, an English, and also a Canadian company made similar interlocking toy bricks and were sued by LEGO for trademark infringement. However, this time, *LEGO* lost. European judges ruled that the eight-peg design of the original LEGO brick "merely performs a technical function and cannot be registered as a trademark." LEGO remains protective of legal issues concerning all their products.

The LEGO MINDSTORMS Evolution

Kjeld Kirk Kristiansen — the current President and CEO of The LEGO Group — was watching children on a TV program using MIT Professor Seymour Papert's Logo programming language to control the behavior of robot turtles. Kjeld asked his management to contact Papert. The LEGO Group visited the MIT Media Lab in Boston, and the idea of computer control for LEGO projects for kids was born in October 1984.

As Kristiansen states: "The philosophy behind LEGO MINDSTORMS is to allow children not only to understand technology, but also to become creative masters of it. This happens when they design, construct, and program their own intelligent inventions."

Seymour Papert

Papert — an MIT mathematician, computer scientist, and educator — is one of the pioneers of artificial intelligence. He is the inventor of the Logo programming language. He created Logo as a tool to improve the way that children think and solve problems. Several companies developed small robots called "Logo Turtles" — one of which (the Tasman Turtle) is shown in **Figure 2**.

These simple robots were used by kids and educators to demonstrate how children could use a computer to control a robot. Many of the turtle robots used a pen or pencil to draw pictures on paper that the robot traversed. Papert also collaborated with LEGO on their Logo-programmable MINDSTORMS kits.

Papert has been called by fellow MIT professor, Marvin Minsky, "the greatest living mathematics educator."

The First Computer-Controlled LEGO Products

Collaboration between LEGO Education and MIT produced LEGO TC Logo in 1986. Children were then able to control the models they built out of different LEGO merchandise. LEGO Technic Control products were based on an interface brick through which students sent signals to LEGO motors and then received information from sensors.

To program their creations, kids used a special version of the Logo computer

Figure 2. The Tasman turtle helps kids learn robot programming through 'Turtle Logo.'

language. LEGO and the MIT Media Lab developed the first prototype of a 'programmable brick,' but their market studies showed the market was not ready at that time since there were so few PCs in homes.

RCX — The First Step in the Evolution of MINDSTORMS

In 1998, the first generation of the MINDSTORMS was the Robotic Invention System. It was regarded as the first "smart toy." Programming the first MINDSTORMS kit required a computer with a serial port. That was not an easy task for many beginners. The RCX (Robotic Command eXplorers) — shown in a close-up in

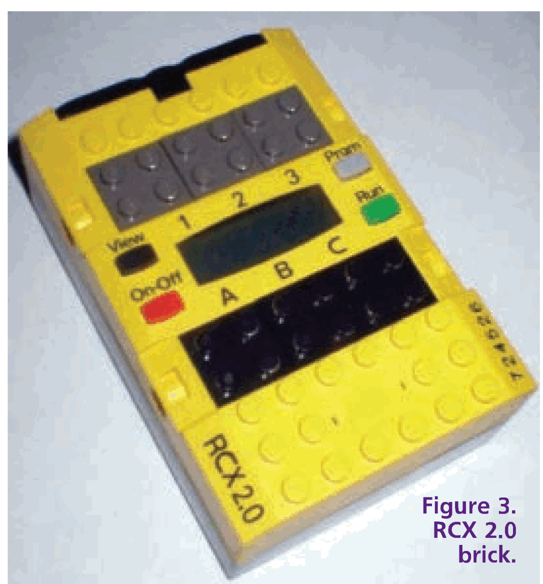


Figure 3. RCX 2.0 brick.

Figure 3 — was the first LEGO microcomputer and was the heart of the first MINDSTORMS system.

The RCX code ran on Microsoft Windows — the most popular PC platform of the time that was available to consumers. The 'brick' contained an eight-bit Hitachi H8/3292 microcontroller as its internal CPU, with 32K of RAM for user programs and LEGO firmware. **Figure 4** from Mr. Alligator shows the LEGO labeled controller chip, with the memory chip above.

The brick was programmed by uploading pre-written programs via an IR interface from a user's PC (and later Macs). The initial version 1.0 brick featured a power jack for external power, whereas the newer version 2.0 has no external jack and must use the internal battery pack (which does not last very long when driving several motors for longer periods).

LEGO introduced MINDSTORMS to the US market on a 30 city ROBOTour '98 Across America. Educators, adults, and children alike were delighted with the RCX brick's amazing capabilities. MINDSTORMS was an instant hit — not only for kids learning the science of robotics, but for college-age students and adults interested in applying microcontroller technology to many applications.

Carnegie Mellon University in Pittsburgh, PA — long known as one of the top universities in the world for robotics education and research — developed the Robotics Academy and ran the very popular summer camp for kids called RoboCamp. Developed in collaboration with LEGO Education, it is designed to teach problem solving, teamwork, and engineering concepts. The new RCX was the ideal centerpiece for this endeavor.

Two or more RCX bricks are able to communicate with each other through the IR interface — an ability that is useful for swarm robots. In addition to the IR interface port, there are three sensor input ports and three motor control output ports that can also be used to drive anything from relays to LEDs.

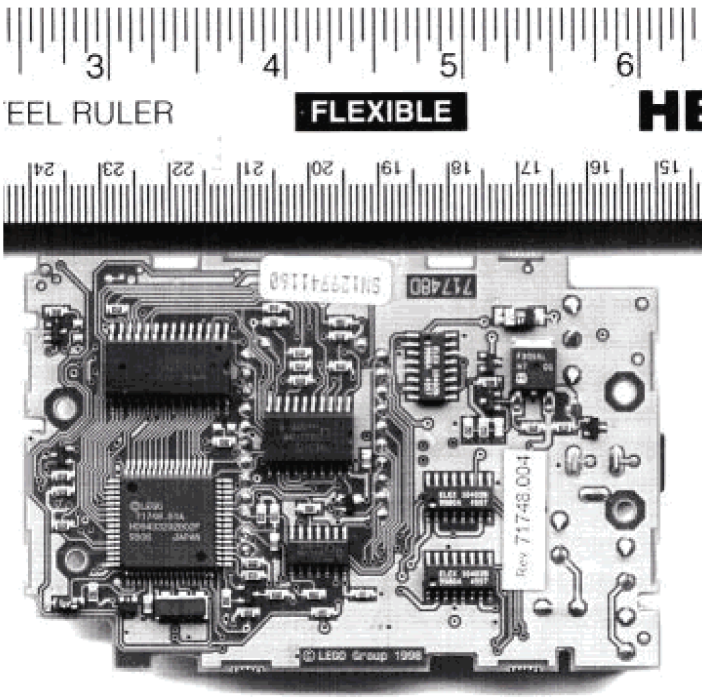


Figure 4. RCX Interior. (Photo by **MrAligator.com**.)

The LCD on the front of the brick can display battery level and input/output port status, as well as which program is selected or running, along with other information. LEGO still provides technical support for this microcontroller brick.

Enter the LEGO MINDSTORMS NXT

In 2006, LEGO introduced its next generation of MINDSTORMS: the NXT Intelligent Brick shown in **Figure 5**. Their 2006 ad campaign stated, "It boasts greater user friendliness" and:

• Expanded sensor capabilities.
• Wireless Bluetooth technology.
• New and improved programming software.
• Progressive curriculum activities.
• Challenges to allow students to come up with new ideas themselves.

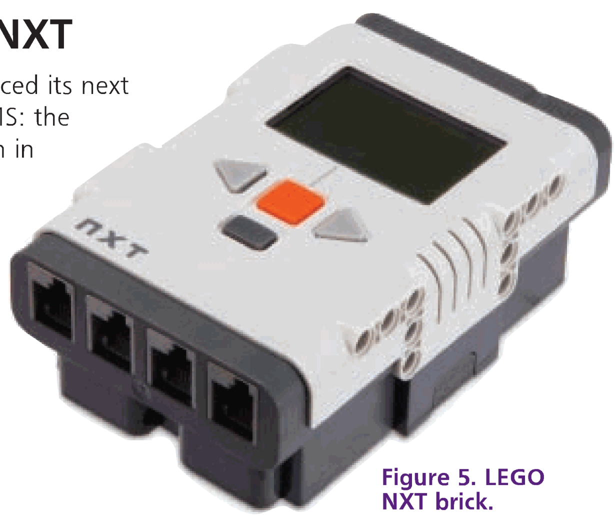


Figure 5. LEGO NXT brick.

"The ROBOLAB software that comes with NXT is powered by National Instruments LabVIEW — a powerful programming software that is used in industry. This program is very visual, having many icons for different commands that the robot can perform."

"At the launch of NXT, Carnegie Mellon's Robotics Academy created 18 weeks of curriculum for schools — Robotics Engineering I and II. Science, technology, engineering, and math (STEM) are addressed in the curriculum and teach students programming basics."

Okay, just what is this new intelligent brick and what makes it better than the older RCX brick?

Figure 6 shows the NXT with three motors and four sensors using cable connections similar to (but not compatible with) common home RJ12 telephone cables. The face of the microcontroller has four momentary touch buttons to control functions and navigate the user interface menus. Information is presented on a

100x60 pixel LCD display that is very adequate to display menus and robot status.

Inside the NXT brick is a 32-bit ARM7 Atmel AT91SAM7S256 microcontroller with 256 KB of Flash memory and 64 KB of RAM, plus a separate eight-bit Atmel AVR ATmega48 RISC microcontroller. The RF link to outside devices is via Bluetooth. The small speaker can play sound files at sampling rates up to 8 kHz.

The kit I have has two power options: the use of six AA batteries (use alkaline) for nine volts; or a 10 volt Li-Ion rechargeable battery pack (the charger is supplied).

Figure 6. NXT brick with motors and sensors.

Figure 7. LEGO MINDSTORMS EV3 intelligent brick.

Figure 9. Self-balancing LEGO robot from the RobotShop blog.

LEGO Sensor Support and Compatibility

Sensors are a very important part of basic and advanced robot design and construction, and I would like to delve into the different types that are available. The LEGO Education Group has recognized this need and has teamed with HiTechnic to provide a wide variety of sensors that are compatible with the NXT and newer EV3 brick (**Figure 7**).

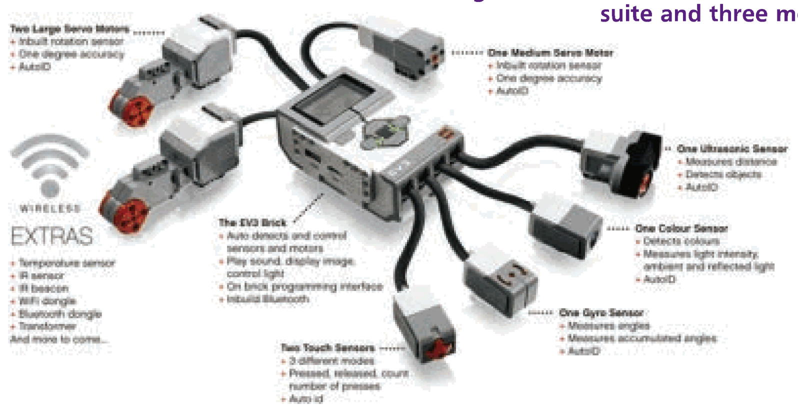


Figure 8. LEGO EV3 with new sensor suite and three motors.

The array of sensors highlighted in **Figure 8** illustrates just how capable a sensor-furnished LEGO robot can become. The new EV3 microcontroller shown cannot only cause motion through up to four electric motors, but can sense its environment via a series of external sensors. Sensors allow a robot to interface with the outside world by detecting objects via an ultrasonic or infrared sensor.

LEGO and HiTechnic have (as part of a kit or as an accessory) over 30 different sensors available for the MINDSTORMS experimenter. These can be viewed at **www.legoeducation.us**. I have selected a few that I have personally used or seen described. Many have the same outward appearance as the accelerometer described next, so I won't show photos of them all. (Please note that some of these sensors are not allowed for use in FIRST LEGO League competition robots.)

• The HiTechnic W991349 acceleration sensor (**Figure 9**) shows a self-balancing robot from the RobotShop blog. It uses the HiTechnic accelerometer module that can detect ±2 g in three axes (X, Y, and Z) with 200 counts per g of acceleration. The acceleration measurement for each axis is refreshed approximately 100 times per second. Notice, also, the ultrasonic sensor acting as the robot's eyes.

• The NXT W979846 ultrasonic sensor (**Figure 10**) shows the NXT version of the sensor; the EV3 is similar in operation with a few advanced specs. They both have an internal microcontroller and can

measure the distance to objects in front of them from 3 cm to 250 cm. The sensor emits a 40 kHz 'chirp' and then measures the reflected echo's return time to determine the distance.

- The EV3 W745509 infrared sensor (**Figure 11**) shows the all new IR sensor that detects within a proximity of 50-70 cm to the robot, and reads signals emitted by the EV3 infrared beacon up to two meters. It can be used in remote-controlled robots to navigate obstacle courses, and you can learn how infrared technology is used in TV remotes, surveillance systems, and even in target-acquisition equipment.
- The HiTechnic W991380 gyro sensor contains a single-axis gyroscopic sensor that detects rotation on the X-Y plane (basic turning), and returns a value that represents the number of degrees per second (±360°) of rotation and the direction of turning.

LEGO has many more sensors that can be applied to advanced robot projects. These include sensors to measure color, a rotational angle sensor to be used as a shaft encoder on wheels or rotating objects, force, sound, light, magnetic fields, temperature, air pressure, pH, and even RFID tags. Most (if not all) of these sensors can be used with both the NXT and the new EV3 bricks, as well as other microcontrollers such as an Arduino, Parallax Propeller, and also laptop computers.

Figure 10. LEGO NXT ultrasonic object and distance sensor.

Figure 11. EV3 infrared sensor.

The LEGO MINDSTORMS EV3

Just what is new with the MINDSTORMS EV3 as compared with the NXT? It is quite a bit more than the stunning new white, black, and red structural theme shown back in **Figure 1**.

The new brick has 64 MB of RAM with 16 MB of Flash memory as compared with 256 KB; the LCD display has 178x128 pixels vs. 100x64, and a faster Atmel ARM 9-SoC processor running open source Linux vs. the ARM 7 on the NXT. The new features include a micro-SD card reader, a USB host port, voice control apps, Apple interconnectivity, Bluetooth v2.1, three-color display backlight, a better speaker, and four motor ports. IR control via a sensor/beacon is also available as a separate purchased item.

The EV3 platform is very compatible with the NXT platform. The building systems are 100% interchangeable, except that the EV3 sensors cannot be used with the NXT brick due to power constraints. The EV3 software is backwards compatible to NXT, and NXT sensors function fully on the EV3 system. It is important to note, however, that the NXT and EV3 bricks cannot be daisy-chained together.

Comparing EV3 with NXT

I wanted to delve a bit into the two previous LEGO MINDSTORMS bricks before examining the new EV3 (the third generation in the series). Considering so much information is available about the NXT, I thought it prudent to not only survey the new features of the EV3 as I did above,

TABLE 1. Comparing EV3 With NXT.

	NXT	EV3
History	Introduced in 2006. Second generation of LEGO MINDSTORMS Education.	Introduced in 2013. Third generation of LEGO MINDSTORMS Education.
Building System	LEGO Technic	LEGO Technic
Software	Graphical and icon based. Powered by National Instruments LabVIEW.	Graphical and icon based. Powered by National Instruments LabVIEW.
Hardware	Intelligent brick, five sensors, three motors, hundreds of parts.	Intelligent brick, five sensors, three motors, hundreds of parts.
Curriculum	Multiple options & hundreds of hours of curriculum for mid to high school STEM. Many community-generated support websites and books.	30+ hours of STEM curriculum available fall semester 2013.
Support	One year warranty. Tech support for the life of the product.	One year warranty. Tech support for the life of the product.
Availability	Now, and sold until the end of 2015.	Available fall semester 2013. Sold indefinitely.
Cost per student over 7 years average use	Under $3	Under $3
FIRST LEGO League	Allowed in 2013 season.	Allowed in 2013 season.
FIRST Tech Challenge	Allowed and supported in 2013-2014 season.	Not allowed in 2013-2014 season.
TETRIX & ROBOTC, LabVIEW for LEGO MINDSTORMS	Supported	Support available fall 2013.

but compare important attributes of both systems.

The chart in **Table 1** should help prospective purchasers in their decision-making — whether they are educators, parents, or advanced experimenters. One important fact to understand for potential buyers of the MINDSTORMS EV3 kits is that the educational kits and the 'home' kits contain different structural components for each group. Special software is available to the educator that is not included in the home version.

The MINDSTORMS EV3 for Education and Advanced Robot Building

LEGO has been around a long time. "Fifteen years ago, we were among the first companies to help children use the power of technology to add life-like behaviors to their LEGO creations with the MINDSTORMS platform," said Camilla Bottke, LEGO MINDSTORMS project lead at The LEGO Group. As I mentioned earlier, LEGO's main thrust with the MINDSTORMS EV3 is children's education, but there is an extremely large user group of university-level and adult experimenters.

If I had one negative to mention about LEGO, it is their website. It is geared towards educators, and specific technical information is extremely difficult to locate on the site. Note, however, that **lego.com** is the non-educational site whereas **legoeducation.us** is geared for educators. Serious users will find the use of a browser to locate particular information at other sites more rewarding, in my opinion.

I would absolutely recommend the EV3 or even the NXT that is still being sold by LEGO. At $339.99US (educational set), the EV3 is only $45US more than the NXT but has so much more to offer. The Linux-based EV3 doesn't even need a computer. You can set up basic routines right on the programmable LEGO brick itself, and/or remote control a robot from an Android or iOS smartphone.

The LEGO Education mission statement as a company is: *Enabling every student to succeed — Our ultimate purpose is to deliver experiences through which students can think creatively, reason systematically, and release their potential to shape their own futures.*

Final Thoughts

Are the LEGO MINDSTORMS EV3 robot kits just toys? Sure, they can be used as toys but they range far beyond that moniker. As I mentioned earlier, the educational and toy sets are marketed through two different sites and contain different components and software, because they are aimed at different markets. They are much more than a box of plastic blocks with some motors and a microcontroller to attach to what you build with the pieces.

LEGO's main emphasis has always been education, and in my opinion, they excel at it. Look past MINDSTORMS EV3 as mere toys and just have fun with them. And, while you're having fun, be sure to do a bit of learning.

There is not a person with any sort of technical bent who would not enjoy sitting down with any of the MINDSTORMS EV3 kits and building one of the robots, programming it, and then operating their creation.

I have been into this field for over 40 years, and yet, I have learned a lot building the NXT and the newer EV3 robots. If a product is affordable, fun for all ages, educational, and hackable, what more could you ask for?

As the large card inside the EV3 box states: *"It's not rocket science, but it can be."* **SV**

I would like to personally thank Trisha McDonell, Public Relations Manager (North America) and Brandi Ellerbee, Senior Account Manager for all their help in assisting me in this endeavor.

Continued from page 17

demonstrations and training. Each unit is assembled to order, so product ships out within 5-7 business days after purchase. List price for this product is $1,200. The cRIO is not included.

Mandy, the Telepresence Robot Platform

AndyMark, Inc., is also announcing the availability of Mandy — a telepresence robot platform.

Mandy is an Arduino-based control system with a three-wheel Killough drive platform using 4" dualie omni wheels and PG27 gearmotors. Mandy is built to order and comes assembled. Allow 1-2 weeks for shipment.

Mandy is actually in the development phase of the product creation cycle, but AndyMark is accepting orders for these development units. Their intention with Mandy is to provide her to robot developers, educational institutions, and users who want to add items and features to her open source hardware and software packages.

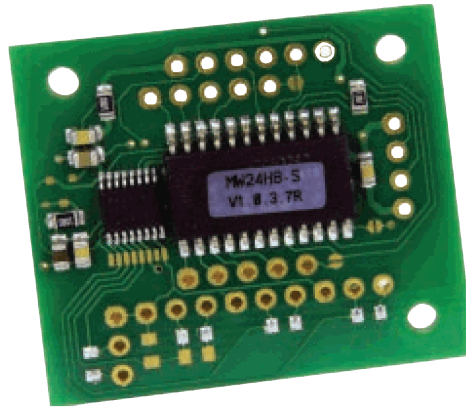
Currently, Mandy is being offered at a beta price of $1,500. Regular price for this unit is $2,500. The beta price is guaranteed through November 14, 2013, and comes with the understanding that AndyMark will be able to offer limited software integration support. The current telepresence application runs through Google Hangout, so minor issues may be encountered when depending on a free communications platform. To help resolve these issues, a Google+ community for Mandy telepresence support has been started to help communicate and address issues. AndyMark is eager to hear back from Mandy customers who might offer suggestions for improvements to this robot platform.

For further information, please contact:

AndyMark, Inc.	Website: **www.andymark.com**

Warrior I/O, Joystick, Mouse USB Modules

Saelig Company, Inc., introduces the new Warrior modules which contain the complete electronics required for joystick, mouse, or I/O functionality via USB.



Based on the successful Warrior USB-interface ICs, these new boards accept analog inputs in the range of 0V to 5V; are compatible with most joystick sensors; and also facilitate measurement, sensor, and other I/O applications.

Warrior ICs from Code Mercenaries are a family of universal I/O controllers for USB that handle all USB's complex protocol details without needing Windows drivers, since Warrior ICs appear to the OS as a HID (Human Interface Device) class. This means that Warrior ICs are controlled without screen prompting for additional software. Standard system drivers allow access to Warrior ICs directly from application programs. (A Linux driver is also available.)

JoyWarrior24A8L-MOD has four analog axes with eight-bit resolution, while JoyWarrior24A10L-MOD has three analog axes with 10-bit resolution. Both modules allow either eight direct connected buttons or 16 buttons in a 4x4 matrix. Four auxiliary outputs can be used to drive LEDs or for other uses. These modules are based on the JoyWarrior24F8 IC — a low cost three-axis acceleration sensor.

MouseWarrior24H8-MOD is a mouse/joystick hybrid which can be switched between mouse and joystick modes at any time. It supports three analog axes with eight bits each and up to six buttons. Auto-centering and drift compensation for the mouse function can optionally be disabled. The MouseWarrior24F8 IC on which this module is based is a mouse replacement chip that does not need an operating surface.

IO-Warrior56-MOD offers 50 general-purpose I/O pins, with a typical maximum input or output rate of 1,000 Hz in a full speed USB2.0-compliant interface (12 Mbit/sec). Based on the IO-Warrior56-LFXI IC, IO-Warrior56-MOD gives simple access to I/O lines via USB since — like other Warrior modules — it has been designed as a generic HID device.

All of the modules work with standard HID system drivers, so no special installation software is needed. Just connect the USB, the axis devices, and buttons, and connection is immediate.

Made in Europe by Code Mercenaries, the Warrior modules are available at a list price of $30.74.

For further information, please contact:

Saelig	Website: **www.saelig.com**

ADVERTISER INDEX


SERVOCITY
We have the parts for your ideas!
Digital and Programmable
HITEC
SERVO POWER GEARBOXES
Build your own custom robot arm!
Channel Mount
Starting at only $74.98 (servo included)
Powerful. Reliable. Affordable.
Order your custom gearbox today!
Top Mount
Bottom Mount
UP TO 3,402 OZ-IN OF TORQUE
Order as a kit or pre-assembled!
Easily mount to channel or tubing
Standard, Continuous and 360° rotation options available
Tube Mount
Brand New!
Actobotics
BALL-BEARING PRECISION BUILDING SYSTEM
DREAM. DESIGN. BUILD. REPEAT.
NEW PRODUCTS ADDED WEEKLY!
Micro Gripper Kit
Aluminum Tubing
.250" Plastic Chain
Drive & Idler Dolly Wheel Plates
Aluminum Servo Arms
Heavy Duty Wheels
GoPro Mount Kit
1/4" Servo Shaft Coupler
Swivel Motor Mounts
Motor Mount D
Base Plate
Ball-Bearing Hub
Servo Plate C
Tube Clamp A
...and much more!
JOIN OUR ONLINE COMMUNITY!
Shop Online | Call 620.221.0123 | sales@servocity.com
www.SERVOCITY.com